ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

Кафедра математического и программного обеспечения информационных систем

АВТОМАТИЗИРОВАННАЯ СИСТЕМА КОНТРОЛЯ ПОСЕЩАЕМОСТИ СТУДЕНТОВ “ШЕБЕКИНСКИЙ ТЕХНИКУМ ПРОМЫШЛЕННОСТИ И ТРАНСПОРТА”

Выпускная квалификационная работа
обучающегося по направлению подготовки 02.03.01 Математика и компьютерные науки, очной формы обучения, группы 07001403
Свешникова Романа Витальевича

Научный руководитель
доцент
Чашин Ю.Г.

БЕЛГОРОД 2018

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

На современном этапе развития образования в России стоит вопрос об особенностях ведения успеваемости и контроля посещаемости студентами учебные заведения.

Актуальность выбранной темы обусловлена тем, что учебная посещаемость, под которой мы понимаем систему присутствия учащихся на занятиях в целях усвоения образовательной программы, практически всегда была важной проблемой образовательного процесса. Учебная посещаемость относится к числу тех категорий, которые, с одной стороны, служат объектом управления

образовательным процессом, с другой - во многом предопределяют образованность и воспитанность детей и молодежи. Учебная посещаемость как явление социально-педагогическое детерминирует академическую успеваемость и воспитание учащихся. Она тесно связана с решением ряда организационных задач учебного процесса, с решением нравственных и социальных проблем семьи и образовательного учреждения, оказывает существенное влияние на их связь между собой, а также на создание имиджа образовательного учреждения и признание высокой квалификации педагогических работников.

Посещение учебных занятий - одна из обязанностей учащихся. Недобросовестное ее выполнение сопряжено с рядом проблем самих учащихся, образовательного учреждения и общества. К учреждениям, где проблема посещаемости периодически становится острой, относятся учреждения начального профессионального образования, центры творчества и другие организации в системе дополнительного образования, школы крупных городов и сельские школы. Педагогический мониторинг учебной посещаемости превращается в важное средство управления образовательным процессом.

Цель работы - Изучение оптимального способа контроля посещаемости студентов, как педагогами, так и родителями, а так же реализация автоматизированной системы, которая позволит упростить данный процесс.

В соответствии с поставленной целью необходимо решить следующие **задачи**:

1. Рассмотреть порядок ведения контроля посещаемости студентов в Шебекинском техникуме промышленности и транспорта;

2. выявить аспекты в порядке ведения контроля посещаемости, которые можно автоматизировать и систематизировать;

3. разработать информационную систему, отражающую структуру подразделения по контролю посещаемости;

4. реализовать ряд Windows приложений для автоматизации ведения посещаемости;

5. реализовать Web-приложение, которое осуществит доступ к информации о посещаемости родителям и педагогам.

Объект исследования - является областное государственное автономное профессиональное образовательное учреждение “Шебекинский техникум промышленности и транспорта”, основным видом деятельности которого является предоставление образовательных услуг.

Предмет исследования - посещения или отсутствия студентов в учебном заведении в часы, когда они должны присутствовать по учебному расписанию.

Выпускная квалификационная работа состоит из введения, четырех глав, заключения, списка используемых источников, приложений.

В первой главе раскрываются основные этапы контроля посещаемости студентов Шебекинского техникума промышленности и транспорта без использования автоматизированной системы. Так же в данной главе приводится пример уже существующих аналогов контроля посещаемости.

Во второй главе продемонстрированы этапы проектирования базы данных, которая будет являться основой для дальнейший разработок приложений.

В третьей главе продемонстрированы этапы реализации Windows приложений для контроля посещаемости, а так же Web-приложения для контроля посещаемости родителями и педагогами.

В четвертой главе проводится тестирование всех реализованных Windows приложений, а так же Web-приложения для контроля посещаемости родителями и педагогами.

ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ОРГАНИЗАЦИИ КОНТРОЛЯ ПОСЕЩАЕМОСТИ В УЧЕБНЫХ ЗАВЕДЕНИЯХ

1.1. Организация учета и контроля посещаемости студентов в Шебекинском техникуме промышленности и транспорта

В настоящий момент во многих учебных заведениях вопрос об электронном контроле посещаемости остается актуальным. Учет и контроль посещаемости обучающимися учебных занятий осуществляется с целью обеспечения максимальной эффективности учебного процесса, совершенствования индивидуальной и самостоятельной работы обучающихся. Однако, силу недостатка ресурсов или некомпетентности руководителей образовательных учреждений, контроль посещаемости выполняется полностью человеческими силами. Очевидно, что такой способ контроля отнимает достаточно времени у сотрудников образовательных учреждений. Ниже будет представлен пример контроля посещаемости в ОГАПОУ “Шебекинском техникуме промышленности и транспорта”.

Для учета и контроля посещаемости учебных занятий в ОГАПОУ ШТПТ ведутся:

- журнал учета посещаемости учебной группы заполняемый старостой;
- ежемесячная ведомость учета посещаемости, которую ведет учебная часть института

На основании ежемесячных ведомостей учета посещаемости учебная часть института по окончании семестра формирует сводную ведомость по каждому направлению подготовки (специальности) и предоставляет её в учебный отдел в течение первой недели экзаменационной сессии.

Сводная ведомость является основой для анализа посещаемости студентов, который проводит учебная часть техникума. В анализе посещаемости студентов отражаются причины низкой посещаемости и разработанные мероприятия по её повышению.

Контроль за посещением учебных занятий обучающимися, правильностью заполнения журналов учета посещаемости старостами групп осуществляет учебная часть института. Контроль за реализацией мероприятий по повышению посещаемости осуществляет проректор по учебно-методической работе.

Журнал учета посещаемости учебной группы является основным академическим документом и предназначен для текущего контроля посещаемости обучающихся. Журнал учёта посещаемости учебной группы и ведомости учёта посещаемости хранятся в учебной части техникума согласно номенклатуре дел.

Ежедневно перед началом учебных занятий староста группы получает журнал учета посещаемости учебной группы в учебной части института и сдает его по окончании занятий. Контроль за правильностью заполнения журналов учета посещаемости проводится учебной частью один раз в неделю. Староста проставляет отметки о присутствии/отсутствии обучающихся на занятии и в конце занятия предоставляет журнал на подпись преподавателю. Если обучающийся отсутствовал, то в журнале ставится отметка «н». Отсутствие обучающегося по уважительной причине (болезнь, вызов в военкомат, участие в соревнованиях, олимпиадах, конференциях, форумах и т.д.) должно быть подкреплено подтверждающим документом, который предоставляется в учебную часть института.

В конце месяца учебная часть техникума высчитывает количество пропусков обучающихся, в том числе по уважительной причине, и заполняет

ежемесячную ведомость учета посещаемости учебной группы, проставляя значения в соответствующие колонки ежемесячной ведомости учета посещаемости. Ведомость учета посещаемости за текущий месяц учебная часть техникума по окончании периода размещает на информационном стенде техникума.

Заместитель директора института по учебной работе, заместитель директора института по внеучебной работе со студентами, куратор академической группы проводят ежемесячный анализ посещаемости обучающимися учебных занятий, устанавливают связь с обучающимися, имеющими большое количество пропусков занятий без уважительной причины и, при необходимости, доводят информацию до родителей обучающихся.

Староста группы несет ответственность за:

- ведение журнала учета посещаемости учебных занятий и его своевременное предоставление на подпись преподавателю;
- восстановление журнала учета посещаемости и успеваемости группы в случае утраты или порчи.

Преподаватель несет ответственность за:

- достоверность информации в журнале групповых занятий преподавателя;
- информирование учебной части о регулярных пропусках студентов.

Учебная часть института несет ответственность за:

- правильность заполнения и хранение журнала учета посещаемости группы;
- ежемесячное составление ведомости учета посещаемости;

- составление сводной ведомости учета посещаемости за текущий семестр и её своевременное предоставление учебному отделу;
- проведение анализа посещаемости студентов.

Директор института несет ответственность за:

- Уровень посещаемости обучающимися занятий;
- выполнение плана мероприятий по повышению посещаемости, разработанному учебной частью института по результатам анализа посещаемости.

Целью данной выпускной квалификационной работы является перевод работы по контролю посещаемости студентов с большого количества человек на электронную систему, которая облегчит задачу педагогов и других сотрудников учебного заведения в анализе посещаемости обучающихся, а так же позволит родителям и кураторам получать своевременно информацию о пропусках их детей и подопечных.

Стоит понимать, что подобную систему невозможно автоматизировать полностью. Работник образовательного учреждения является обязательным звеном в работе системы, так как имеется возможность ложных отметок в журнале. Каждый прогул студента должен контролироваться уполномоченным работником учебной части, а так же сопровождаться комментарием. Это необходимо, чтобы вести историю прогулов и чтобы была возможность восстановить причину пропуска даже через большой отрезок времени.

1.2. Анализ информационных систем для контроля посещаемости

В настоящее время существует множество систем для контроля посещаемости и не только в учебных заведениях. Как пример, существуют система контроля и управления доступа (СКУД). Основная задача любой СКУД – это обеспечение безопасности рабочего персонала и сохранности ценного имущества. В отличие от охранных сигнализаций, такие системы позволяют не просто наблюдать за обстановкой на территории предприятия, но и ограничивать доступ к тем или иным помещениям. Перечень возможностей СКУД позволяет организовать контроль посещаемости студентов, однако такая система обременена излишними функциями, которые никак не будут задействованы в требуемой системе:

• Учет посещаемости помещений и внешних территорий. Данная возможность позволяет отслеживать все перемещения людей и служебного транспорта на предприятии. При необходимости пользователь может обратиться к записям из архива посещений, и идентифицировать людей, которые находились на территории того или иного объекта в конкретное время (например, для выяснения обстоятельств аварии на производстве, кражи или других инцидентов);

• ограничение доступа в помещения. Эта функция дает возможность контролировать посещение отдельных помещений и территорий, и тем самым предотвращать проникновение посторонних лиц на объекты с ограниченным доступом (кабинеты бухгалтерии и начальства, офисы, склады с ценными товарами и пр.);

• детализированный расчет зарплаты персонала. Система контроля и управления доступом, будучи синхронизированной со специальной программой бухучета, позволяет владельцу предприятия с высокой точностью рассчитывать количество рабочих часов каждого сотрудника, и выплачивать зарплату по факту. Это дает возможность сэкономить денежные средства из

фонда компании, и предотвратить лишние расходы на оплату труда рабочих, которые регулярно опаздывают или покидают свой пост до окончания рабочего дня;

• синхронизация с системой видеонаблюдения. Объединив СКУД и систему охранного видеонаблюдения, владелец предприятия получает возможность быстро идентифицировать автотранспорт, проезжающий на охраняемую территорию, а также фиксировать всех посетителей тех или иных помещений.

Как видно из вышеперечисленных функций, ни одна не требуется в задаче контроля посещаемости студентов образовательного учреждения.

Так же существует система контроля посещаемости, которая приближена к теме образовательных учреждений - RFID-система, где каждый сотрудник и ученик получает бейдж или браслет со специальной RFID-меткой, содержащей определённую информацию об объекте. Ученикам датчики можно установить на дневник, чтобы он одновременно был и документом школьника и его пропуском на занятия. Метку можно считать мобильным или стационарным считывателем, после чего вся информация попадает в единую компьютерную базу.

RFID-система дает следующие возможности:

• Регистрировать события такие как время входа, выхода, нахождения человека в зоне, к примеру, когда человек зашёл в помещение, сколько пробыл, когда вышел;

• контроль действий персонала: все перемещения и действия учителей, уборщиков, поваров, медицинского персонала и охранников находятся под наблюдением. Ведётся учёт операций каждого сотрудника;

• контроль действий учащихся: способ борьбы с прогулами и опозданиями. Для каждого ученика существует индивидуальная учётная запись, в которой ведётся мониторинг всех его передвижений.

Вновь возможности RFID-системы обширны, но нужна будет их малая часть. Так же стоит учесть, что системы RFID и СКУД весьма тяжелые в развертывании на объекте, требуют дополнительного обученного персонала, а так же требуют выделения большого бюджета на содержание и установку. Образовательные учреждения в большинстве содержатся на федеральном бюджете, поэтому мало кто может себе позволить такие системы. Этим и объясняется, что в техникумах, школах и университетах Российской Федерации системы контроля посещаемости не имеют большого распространения.

К сказанному в конце главы 1.1 можно добавить, что целью выпускной квалификационной работы является не только реализация системы контроля посещаемости, а так же должны быть учтены финансовые возможности предприятия по внедрению автоматизированной системы. Необходимо, чтобы управление системой не требовало специальных навыков от сотрудников образовательного учреждения.

ГЛАВА 2. ПРОЕКТИРОВАНИЕ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ КОНТРОЛЯ ПОСЕЩАЕМОСТИ СТУДЕНТОВ

2.1. Проектирование информационного обеспечения

2.1.1. Выбор СУБД

При выборе СУБД следует в первую очередь определиться с двумя основными критериями, влияющими на эффективность их использования в качестве эффективного хранилища данных: количество одновременно работающих пользователей и объем ежедневно вносимой в базу данных информации. В терминах СУБД эти критерии относятся соответственно к механизму реализации так называемых «блокировок» и механизму «секционирования» данных для уровня изоляции транзакций «readcommited», используемому Fansy-приложениями по умолчанию. Механизм «блокировок» обеспечивает гарантию целостности считывания и изменения данных каждым пользователем при многопользовательском режиме доступа. Важным фактором при выборе платформы следует считать тот аспект, блокирует ли СУБД данные таблиц при вставке записей одними пользователями, для тех пользователей, которые используют эти таблицы для чтения. В типовой ситуации одни пользователи проводят операции текущего дня, в то время как другие хотят получить отчеты за прошедший период, при этом используются данные одних и тех же таблиц. При выборе СУБД для многопользовательского использования предпочтение следует отдавать СУБД, которые не блокируют соседние записи при вставке и изменении данных.

Механизм «секционирования» данных (partitioning) позволяет разделять большие (критичные по объему) базы данных, таблицы и индексы на множество мелких управляемых частей, каждая из который может быть доступна более эффективным способом, чем весь массив данных в целом. Например, разнесение данных и индексов на два параллельных диска позволят ускорить доступ к данным, т.к. чтение индексов не скажется на производительности чтения данных. Секционирование большой таблицы, индексируемой по дате, на несколько таблиц по интервалу дат, позволит получить более быстрый доступ к данным текущего отчетного периода, при этом доступ к данным предыдущих отчетных периодов замедлится только в случае получения консолидированных данных по периодам, относящихся к разным секциям хранения.При выборе СУБД для хранения больших массивов данных предпочтение следует отдавать СУБД, которые имеют гибкий механизм секционирования.

Еще одним критерием являются требования к аппаратному обеспечению и квалификации персонала для обеспечения безопасности и эффективности функционирования СУБД.

Сравнение существующих СУБД по вышеперечисленным критериям приведен в таблице 2.1.

Таблица 2.1

Сравнение разных СУБД по критириям

Критерий	FIREBIRD	MySQL	ORACLE
Наличие блокировок чтения соседних строк при вставке и изменении данных?	Нет блокировок, дополнительные ресурсы не требуются	Блокировки есть. Эскалация блокировок может привести к блокированию таблицы в целом. Блокировок нет только при включении специального уровня изоляции SNAPSHOT.	Нет блокировок, дополнительные ресурсы не требуются

Разделения данных пользователя, индексов и системных данных?	Нет разделения	Есть разделение	Есть разделение

Таблица 2.1 (продолжение)

Критерий	FIREBIRD	MSSQL	ORACLE
Наличие секционировани я больших таблиц.	Нет секционировани я	Есть секционирование толь ко при создании таблицы (секционированные таблицы ограничены по структуре индексов)	Есть секционирование как при создании, так и при модификации любых таблиц
Требования к аппаратному обеспечению	Нет особых требований, база занимает один файл на диске	Желательно наличие нескольких параллельных дисков для разделения пользовательских областей данных, системных областей данных и журнала транзакций	Требуется наличие нескольких параллельных дисков (не менее 3-х) для функционирования СУБД и желательно наличие дополнительных параллельных дисков для секционирования данных пользователя
Требования к квалификации персонала?	Нет особых требований. Достаточно знаний о процедуре дампа баз данных, для обеспечения их сохранности.	Требуются знания о процедуре дампа баз данных, для обеспечения их сохранности.Требуются знания о командах DDL на секционирование данных и перестройке индексов.	Требуются квалификация в настройке процесса обеспечения отката транзакций и в мапировании табличных пространств на различные дисковые пространства.Требуют ся знания о командах DDL на секционирование данных.

В данной выпускной квалификационной работе используется СУБД FireBird и это решение было принято так же с учетом таких фактов:

- Простота в использовании и администрировании;

- FireBird поддерживает многоверсионную архитектуру, которая позволяет иметь долгие по времени существования транзакции;
- СУБД FireBird не имеет привязки к конкретной ОС; СУБД FireBird бесплатна и имеет открытый исходный код. [2]

Однако имеются и некоторые недостатки:

- СУБД FireBird проигрывает в производительности многим другим СУБД на операционной системе Windows;
- СУБД FireBird подходит для администрирования баз данных небольшого объема; [2]
- нет явного способа сменить владельца базы данных.

Основываясь на вышеперечисленных аргументах, было принято решение использовать СУБД FireBird, так как для автоматизированной системы не будет создаваться БД большого объема и не маловажно то, что СУБД бесплатна.

2.1.2. Системный анализ предметной области

Первым этапом проектирования информационной системы является анализ предметной области. На данном этапе важно проанализировать потребности пользователей, выбрать информационные объекты и обозначить их характеристики для определения содержимого информационной системы. На основе проведенного анализа структурируется предметная область.

От студентов необходимо знать их ФИО, номера телефонов и номера телефонов родителей.

Множество студентов объединяются в учебные группы, которые потом посещают занятия, согласно расписанию именно группы, а не студента.

В расписание входит много параметров, таких как:

- Порядковый номер недели;
- день; пара;
- дисциплина;
- группа;
- кабинет;
- дата.

Порядковый номер недели определяет, какая неделя в семестре. Это необходимо, чтобы отличать одну неделю от другой.

От других информационных объекты, как кабинет, дисциплина, пара, группа требуется знать лишь их наименование.

Для успешного ведения посещаемости необходимо знать расписание студентов. Расписание состоит из пар. Связав конкретную пару и студента, можно сделать отметку о посещении этого студента данной пары. Таким образом, главные зависимости в системе контроля посещаемости это расписание(пары) и студенты. В дальнейшем, связь студент-пара будет именоваться ведомость посещаемости.

2.1.3. Инфологическое проектирование

Цель инфологического этапа проектирования состоит в получении семантических (концептуальных) моделей, отражающих предметную область и информационные потребности пользователей. В качестве инструмента для построения семантических моделей данных на этапе инфологического проектирования является неформальная модель "Сущность-Связь" (ERмодель - Entity-Relationship). Моделирование предметной области базируется на использовании графических диаграмм, включающих небольшое число разнородных компонентов.

Основными понятиями ER-модели являются сущность, связь и атрибут.

Сущность (объект) - это реальный или представляемый объект предметной области, информация о котором должна сохраняться и быть доступна. Различают такие понятия, как тип сущности и экземпляр сущности. Понятие тип сущности относится к набору однородных предметов, событий, личностей, выступающих как единое целое. Экземпляр сущности относится к конкретной вещи в наборе. В диаграммах ER-модели сущность представляется в виде прямоугольника (в нотации Баркера), содержащего имя сущности.

Атрибут - поименованная характеристика сущности, определяющая его свойства и принимающая значения из некоторого множества значений.
Каждый атрибут обеспечивается именем, уникальным в пределах сущности.

Атрибуты могут классифицироваться по принадлежности к одному из трех различных типов: описательные, указывающие, вспомогательные. Описательные атрибуты представляют факты, внутренне присущие каждому экземпляру сущности. Указывающие атрибуты используются для присвоения имени или обозначения экземплярам сущности. Вспомогательные атрибуты используются для связи экземпляра одной сущности с экземпляром другого.
Атрибуты подчиняются строго определенным правилам.

Множество из одного или нескольких атрибутов, значения которых однозначно определяют каждый экземпляр сущности, называются идентификатором. Каждый экземпляр сущности должен иметь хотя бы один идентификатор. Если идентификаторов несколько, один из них выбирается как привилегированный.

Связь (Relationship) - это поименованная графически изображаемая ассоциация, устанавливаемая между сущностями и представляющая собой абстракцию набора отношений, которые систематически возникают между

различными видами предметов в реальном мире. Большинство связей относятся к категории бинарных и имеют место между двумя сущностями.

Среди бинарных связей существуют три фундаментальных вида связи: один-к-одному (1:1), один-ко-многим (1:M), многие-ко-многим (M:M). Связь один-к-одному (1:1) существует, когда один экземпляр одной сущности связан с единственным экземпляром другой сущности. Связь один-комногим (1:M) имеет место, когда один экземпляр одной сущности связан с одним или более экземпляром другой сущности и каждый экземпляр второй сущности связан только с одним экземпляром первой сущности. Связь многие-ко-многим (M:N) существует, когда один экземпляр одной сущности связан с одним или более экземпляром другой сущности и каждый экземпляр второй сущности связан с одним или более экземпляром первой сущности.

На этапе инфологического проектирования необходимо определить связи между сущностями, а также описать ограничения целостности, то есть требования к допустимым значениям данных и к связям между ними.

Основываясь на информации, полученной в прошлом пункте, можно сделать выводы:

Объект «Студент» является подмножеством объекта «Группа». В одной группе может быть не один студент, но студент не может числиться сразу в нескольких групп. Это значит, что связь «Студент»-«Группа» имеет тип «один-ко-многим».

Так же, объекты «День», «Неделя», «Дисциплина», «Кабинет», «Порядок пары», «Группа» являются подмножеством объекта «Расписание». Как и в прошлом случае, подобная связь имеет тип «один-ко-многим».

Чтобы вести учет посещаемости студентов образовательного учреждения, необходимо связать объекты «Студент» и «Расписание», однако, студент может пропустить несколько пар и в тоже время, одну пару могут

пропустить несколько студентов. Такой тип связи называется «многиеко-многим» и чтобы его реализовать, необходимо ввести новый объект – «Пропуск».

Все сущности связаны в единую модель информационной системы, определены типы связей между сущностями, описаны ограничения целостности. Инфологическое проектирование завершено. На основе полученных данных можно составить логическую схему базы данных, которая изображена на рис. 2.1.

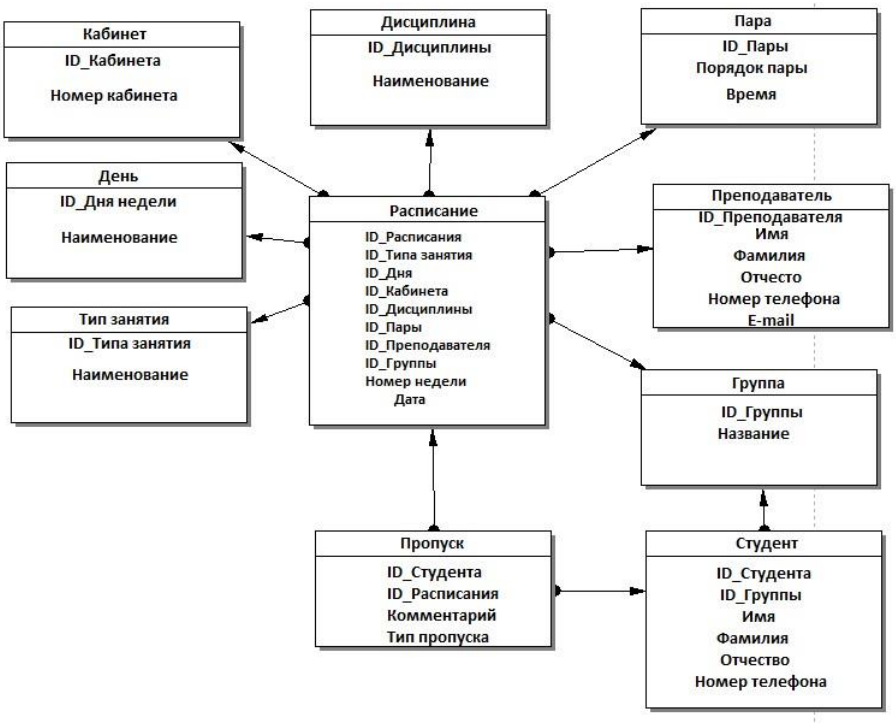


Рис. 2.1. Логическая схема базы данных

2.1.4. Даталогическое проектирование

Логическое (даталогическое) проектирование — создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели данных даталогическая модель — набор

схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи. В ходе даталогического проектирования, основываясь на результатах инфологического проектирования, необходимо выполнить нормализацию модели информационной системы. Нормализация используется для устранения из сущностей нежелательных функциональных зависимостей, которые порождают возможные противоречия хранимых данных, избыточность данных, аномалии ввода, обновления, удаления, нарушают целостность данных.

Обычно нормализация делится на три этапа:

1. Приведение сущностей в 1 нормальную форму. Сущность находится в первой нормальной форме, если значения всех ее атрибутов атомарны;

2. приведение сущностей во 2 нормальную форму для исключения аномалий, возникающих в отношениях, находящихся в 1 нормальной форме. Сущность находится во второй нормальной форме, если она находится в 1 нормальной форме, а каждый ее не ключевой атрибут функционально и полно зависит от ключа. Вторая нормальная форма требует, чтобы не было не ключевых атрибутов, которые зависят только от части первичного ключа;

3. приведение сущностей в 3 нормальную форму для исключения аномалий, возникающих в отношениях, находящихся во 2 нормальной форме. Сущность находится в третьей нормальной форме, если она находится во второй нормальной форме и все не ключевые атрибуты зависят только от первичного ключа.

Перейдём к нормализации модели полученной в ходе инфологического проектирования (рис. 2.1).

Внимательно изучив каждую из сущностей модели, можно отметить, что значения всех атрибутов атомарны. Делаем вывод, что все сущности модели находятся в 1 нормальной форме.

Далее, проанализировав зависимости неключевых атрибутов сущностей от ключа, видим, что каждый из них функционально и полно зависит от ключа, то есть все сущности модели находятся во второй нормальной форме.

Также можно видеть, что все неключевые атрибуты сущностей зависят только от первичного ключа и не зависят друг от друга.

Подводя итог, приходим к вывод, что модель информационной системы, полученная в ходе инфологического проектирования, находится в третьей нормальной форме.

2.2. Проектирование программного обеспечения

2.2.1. Выбор средств разработки

Для решения цели создания Windows приложения, предназначенного для работы с созданной базой данных под управлением СУБД Firebird, была выбрана среда разработки Borland C++ Builder 6. Данная среда разработки приложений имеет все необходимые инструменты и компоненты для создания полнофункционального приложения, осуществляющего работу с базой данных.

Так же C++ Bilder имеет готовые компоненты по работе с СУБД FireBird, которые показаны на рис. 2.2.

Рис. 2.2. Компоненты InterBase в C++ Bilder

Список компонентов, которые есть в C++ Bilder:

- iBTable (набор данных Table);
- iBQuery (набор данных Query);
- iBstoredProc (вызов хранимой процедуры);
- iBDatabase (соединение С БД);
- iBTransaction (транзакция);
- iBUpdateSQL (изменение набора данных, основанного на SQL-запросе);
- iBDataset (источник данных);
- iBSQL (выполнение SQL-запроса);
- iBDatabaseinfo (информация о БД);
- iBSQLMonitor (монитор выполнения SQL-запросов);
- iBEvents (событие сервера);
- iBExtract (извлечение данных);
- iBClientDataSet (клиентский источник данных).

Для решения цели создания Web приложения, предназначенного для работы с базой данных под управлением СУБД Firebird, был выбран язык программирования, специально нацеленный на создание Web приложений - PHP версии 5.3 с поддержкой технологии доступа к базам данных PDO. Выбор технологии PDO обоснован тем, что для работы с различными СУБД необходимо только изменить параметры подключения в зависимости от выбранной базы данных. Остальной программный код остается без изменений.

2.2.2. Определение задач, решаемых информационной системой

Для решение главной задачи – контроль посещаемости студентов образовательного учреждения, система должна состоять из трех Windows приложений и одного web-приложения.

Windows приложение для администратора должно выполнять следующие задачи:

- Добавление записей в базе данных без входа в СУБД;
- изменение существующих записей по признаку;
- каскадное удаление записей;

Windows приложение для создания расписания должно выполнять следующие задания:

- Создание расписания;
- редактирование уже существующего расписания группы;
- удаление расписания группы;
- очистка полей ввода;
- копирование расписание одной недели и вставка его на позиции остальных недель.

Windows приложение для ведения посещаемости должно выполнять следующие задачи:

- Добавление меток о прогулах студента;
- редактирование уже существующих меток о прогулах студента;
- прикрепление к метке сопроводительного комментария о прогуле студента;
- поддержка динамических объектов, так как заранее системе не известно, сколько студентов в группе.

Web-приложение контроля посещаемости для родителей студентов должно выполнять следующие задачи:

- Демонстрация посещаемости студенческой группы по выбору даты;
- демонстрация индивидуальной ведомости посещаемости студента, собранной по месяцам;

ГЛАВА 3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ КОНТРОЛЯ ПОСЕЩАЕМОСТИ СТУДЕНТОВ

3.1. Разработка информационного обеспечения

3.1.1. Физическая модель базы данных

Физическое проектирование — создание схемы базы данных для конкретной СУБД. Специфика конкретной СУБД может включать в себя ограничения на именование объектов базы данных, ограничения на поддерживаемые типы данных и т. п. Кроме того, специфика конкретной СУБД при физическом проектировании включает выбор решений, связанных с физической средой хранения данных (выбор методов управления дисковой памятью, разделение БД по файлам и устройствам, методов доступа к данным), создание индексов и т. д.

Основываясь на результатах, полученных в ходе инфологического и даталогического проектирования, создадим физическую модель информационной системы для СУБД Firebird. Для этого необходимо:

- Установить первичные и внешние ключи сущностей для сохранения целостности данных;
- установить типы данных ключей;
- установить типы данных атрибутов сущностей;

Первичным ключом сущности «Студент» является идентификационный номер студента. Внешние ключи данной сущности не требуются, так как данная сущность связана только с сущностью «Группа».

Первичным ключом сущности «Дисциплина» является идентификационный номер дисциплины.

Первичным ключом сущностей «Преподаватель», «Пара», «Кабинет», «Группа», «Тип занятий», «День» является идентификационный номер пары, кабинета, группы и типа занятий соответственно.

Первичным ключом сущности «Расписание» является идентификационный номер расписания. Внешними ключами сущности «Расписание» являются идентификационные номера:

- Преподавателя;
- кабинета;
- дисциплины;
- группы;
- типа занятия;
- пары; дня.

Так же стоит помнить об особенной сущности «Пропуск», которая не имеет уникального номера, так как с помощью этой сущности формируется связь типа «многие-ко-многим». У сущности «Пропуск» существуют только два внешних ключа – идентификационный номер расписания и идентификационный номер студента.

Типы данных первичных и внешних ключей установим как smallint. Этот тип, использующий целые числа, позволяет хранить до 32676 значений, что достаточно для реализации базы данных контроля посещаемости. Также причиной использования этого типа является небольшой объем 2 байта, а также то, что этот тип представлен во множестве других СУБД. Надо учитывать, что в процессе эксплуатации системы контроля посещаемости, может быть применены изменения, такие как переход на другую СУБД. В

дальнейшем все устанавливаемые типы данных будут выбираться с учетом этого критерия.

Для атрибутов, таких как имя, фамилия, отчество будем использовать тип данных varchar(15) . Двадцати символов вполне достаточно для каждого элемента ФИО как русских, так и для иностранных имен. Так же, стоит отметить, что не у всех есть отчество. Необходимо это учесть. Такой же тип данных подойдет и для номера телефона.

Для атрибутов, где требуется ввод большего объема текста будет использоваться тип данных varchar(50) и varchar(255). Последний тип подойдет для хранения сопроводительных комментариев к отметке о прогулах студентов. Varchar(50) – тип данных, который подойдет к названию дисциплин и электронной почты преподователей.

Подводя итог, можно составить физическую схема базы данных контроля посещаемости, которая приведена на рис.3.1.

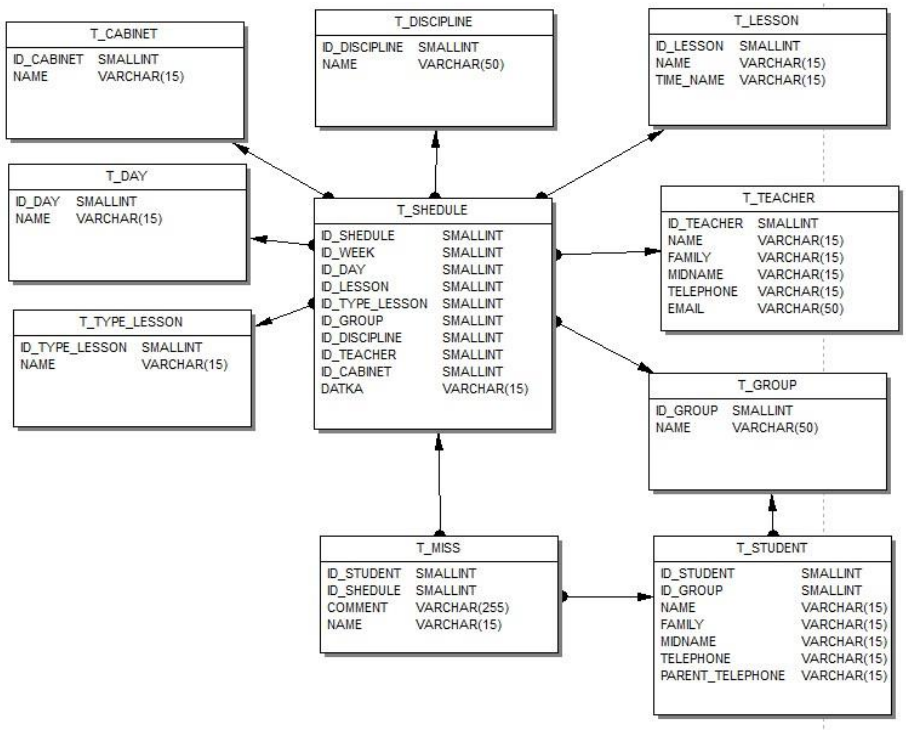


Рис 3.1. Физическая схема базы данных контроля посещаемости

Как видно из рисунка, главная сущность «Расписание» и большинство остальных сущностей лишь ее подмножество, однако, для контроля посещаемости большую роль играет сущности «Пропуск».

3.1.2. Программирование на стороне SQL сервера Firebird

Программирование на стороне SQL сервера включает в себя следующие этапы:

1. Создание домено;
2. создание таблиц;
3. создание триггеров и генераторов.

Для создания доменов требуется определить типы атрибутов сущностей, выявить общие, организовать такие типы в домен, задать название, указать тип, длину, ограничения, значение по умолчанию. Процесс создания доменов показан в таблице 3.1.

Таблица 3.1

Основные домены в базе данных

Имя домена	Тип	Not Null	Default	Check
D_KEY	smallint	X	0	---
D_INT	integer	X	0	---
D_LONG_TEXT	varchar(255)		Не определен	---
D_NAME	varchar(15)		---	---
D_SMALL_TEXT	varchar(50)		---	---
D_DATE	date	X	Текущая дата	>=current_date()

Как видно из таблицы 3.1.2.1, большинство использованных типов это типы для работы со строками. Это обусловлено потребностью в большом количестве атрибутов, которые являются последовательностями символов.

Однако, среди созданных доменов есть домен «D_DATE», который позволяет хранить информацию о дате. Можно было в качестве даты использовать строковый тип, но тогда бы пришлось в процессе разработки приложений очень часто использовать функции для нахождения подстрок в строке, так как нет возможности из строковой переменной получить значение месяца, например. Домен «D_DATE», который имеет тип date универсален и работа с ним реализована во многих средствах разработки приложений.

Так же стоит заметить, что не все домены позволяют хранить пустое значение. Как пример, домен «D_KEY» является доменом всех ключевых полей. Пустое ключевое поле при добавлении записи недопустимо, так может повлечь за собой сбой работы всей базы данных. Так же и домен «D_DATA» не может быть пустым, так как при добавлении расписания не может отсутствовать дата.

Здесь можно подумать, что дата может быть ключевым полем для расписания, но это ошибка. В один день может быть несколько пар, а значит если для каждой пары будет первичным ключом дата, то получится, что для нескольких записей будет одинаковый ключ, а это нарушение первой формы базы данных.

Ниже будет приведены таблицы, которые отражают сущности в базе данных и типы их полей.

Таблица 3. 2

Типы полей в таблице «Студент»

Поле	Тип	Not Null	Default	Check	Primary key	Имя домена
id_student	SMALLINT	X			X	D_KEY
Id_group	SMALLINT	X				D_KEY
FAMILY	VARCHAR					D_NAME
NAME	VARCHAR					D_NAME

MIDNAME	VARCHAR					D_NAME
TELEPHONE	VARCHAR					D_SMALL_TEXT
PARENT_ TELEPHONE	VARCHAR					D_SMALL_TEXT

В таблице 3.2. можно заметить, что обязательными полями являются только ключевые поля. Это обусловлено тем, что некоторые иностранные граждане не имеют отчества. Так же, поля для номера телефонов студентов и родителей являются необязательными, так как не у всех есть таковые.

В таблице 3.3. приведены типы полей в сущности «Расписание». Так как эта сущность является сводной для всех остальных, все поля обязательны в ней.

Таблица 3.3

Типы полей в таблице «Расписание»

Поле	Тип	Not Null	Default	Check	Primary key	Имя домена
id_shedule	SMALLINT	X			X	D_KEY
id_group	SMALLINT	X				D_KEY
id_teacher	SMALLINT	X				D_KEY
id_lesson	SMALLINT	X				D_KEY
id_discipline	SMALLINT	X				D_KEY
id_type_lesson	SMALLINT	X				D_KEY
id_week	SMALLINT	X				D_KEY

id_cabinet	SMALLINT	X				D_KEY
datka	DATE	X	current_date	>=current_date		D_DATE

В атрибуте «datka» стоит условие проверки. Это сделано для того, чтобы работник учебной части, при создании и редактирования расписания по ошибке не поставил дату в прошлом.

Дальше приведена таблица 3.4, которая показывает все атрибуты и поля в сущности «Пропуск». В этой сущности отсутствует первичный ключ, есть только внешние ключи.

Таблица 3.4

Типы полей в таблице «Пропуск»

Поле	Тип	Not Null	Default	Check	Foreign key	Имя домена
id_shedule	SMALLINT	X			X	D_KEY
id_student	SMALLINT	X			X	D_KEY
name	VARCHAR	X				D_NAME
comment	VARCHAR		“Нет информации”			D_LONG_TEXT

Два внешних ключах в таблице «Пропуск» осуществляют связь «многие-ко-многим». Так же тут присутствует атрибут comment, который отвечает за сопутствующий комментарий при выставлении отметки о прогуле студента. В образовательном процессе зачастую студент не может назвать причину отсутствия на парах, поэтому сотруднику учебной части будет проще пропустить сопутствующий комментарий, а база данных сама занесет в поле значение “Нет информации”.

Собрав воедино всю информацию о сущностях и их атрибутах можно приступить к непосредственной реализации на СУБД FireBird. Реализация доменов приведена на рис. 3.2.

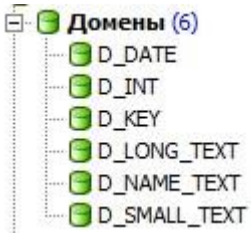


Рис.3.2. Домены, реализованные на СУБД FireBird

Далее, используя созданные домены, а так же информацию из физической схемы базы данных можно приступить к созданию таблиц на СУБД FireBird. Итоговый результат приведен на рис. 3.3.

Когда будет создаваться приложение для администратора базы данных, необходимо реализовать возможность автоинкрементации ключевых полей в таблицах.

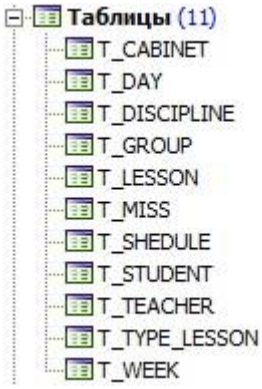


Рис. 3.3. Создание таблиц в СУБД FireBird

Это необходимо для того, чтобы администратор не был обременен заботой о значениях ключевых полей. Ведь может случиться, что какие-то

записи будут удалены, вместе с их значениями ключевых полей. Администратор не должен хранить эти значения. Вместо этого подойдет решение, как создание триггера для каждой таблицы. Когда будет заноситься новая запись, триггер срабатывает и увеличивает значение ключевого поля на единицу, таким образом, удаляется возможность создания записи с повторяющимся значением ключевого поля.

После создания триггеров и генераторов, можно приступить к реализации бизнес-логики базы данных. В это входят следующие этапы:

1. Создание представлений (просмотров);
2. создание хранимых процедур;
3. создание исключений.

Представление в базе данных нужны для вывода информации из базы данных, но с некоторыми условиями. Если выводить полностью таблицу, то может быть получены не понятные для человеку значение. Так, например, если попробовать вывести таблицу «Пропуск», то пользователь вместо ожидаемого формата «Предмет-Студент» получит список пар целочисленных значений. Чтобы такого не происходило, необходимо создать готовые представления, которые по полученным парам целочисленных значений находили название предмета и ФИО студента. Ниже будет приведена таблица 3.5, которая показывает все созданные представления.

Таблица 3.5

Готовые представления в базе данных

Представление	Функция
STUDENT_INFO	Информация о студентах.
TEACHER_INFO	Информация о преподавателях.
MISS_INFO	Информация о пропусках.

GROUP_INFO	Полная информация о группах
SHEDULE_INFO	Вывод расписания всех групп.

Хранимые процедуры в базах данных предназначены для упрощения работы тех, кто будет подключаться в базе данных и вносить изменения. Во многих средствах разработки приложений существуют способы и компоненты для вставки записей, удаления и обновления, однако, такие способы занимают достаточно времени на их реализацию и зачастую в долгосрочной перспективе могут привести к ошибкам. Для предотвращения этого внутри самой базы данных можно создать процедуры для манипуляций с таблицами. Вызов процедур осуществляется минимальным затратам времени, а так же не могут привести к ошибкам в работе с базой данных.

Можно разделить хранимые процедуры на такие категории:

• Добавление записи – создается новая запись, которая добавляется в необходимую таблицу;

• удаление записи по признаку – удаление существующей записи, однако, удаление возможно как минимум с одним параметром. Этот параметр необходим для обнаружения записи в таблице, которую следует удалить;

• изменение записи по признаку – изменение существующей записи, однако, изменение возможно как минимум с одним параметром. Этот параметр необходим для обнаружения записи в таблице, которую следует изменить.

В таблице 3.6. приведены хранимые процедуры в базе данных.

Таблица 3.6

Хранимые процедуры в базе данных

Название процедуры	Функция
ADD_STUDENT	Добавляет студента в группу

ADD_TEACHER	Добавляет преподавателя
ADD_SHEDULE	Добавляет пару в расписание группы
ADD_MISS	Добавляет пропуск студента
DELETE_STUDENT	Удаляет студента из группы
DELETE_TEACHER	Удаление преподавателя
DELETE_SHEDULE	Удаляет пару из расписания группы
DELETE_MISS	Удаление информации о пропуске

Исключения в базе данных необходимы для тех случаев, когда запрос или действие пользователя привели к нежелательным последствиям, либо просто к ошибке. Чтобы не возникало таких ситуаций, создаются готовые исключения в базе данных, чтобы они отображались пользователю, когда происходит ошибка. В базе данных контроля посещаемости студентов образовательного учреждения существует потребность в некоторых исключениях. Список всех исключений приведен в таблице 3.7.

Таблица 3.7

Исключения в базе данных

Название исключения	Функция
WARN_DATE_FORMAT	Ошибка с форматом даты.
WARN_MISS	Ошибка с некорректным созданием пропуска
WARN_DEL_STUDENT	Ошибка при удалении студента из базы данных

На этом реализация бизнес-логики в базе данных завершено. Теперь можно переходить непосредственно к реализации приложений.

3.2. Разработка программного обеспечения

3.2.1. Разработка Windows приложения для администратора

Разработка всех трех Windows приложений включает в себя следующие этапы:

1. Выбор и настройка необходимых компонентов;
2. организация пользовательского интерфейса;
3. программирование компонентов (написание программного кода).

Компоненты для работы с базой данных: DataModule для хранения компонентов, IBDatabase для установления соединения с базой, IBTransaction для осуществления транзакций, IBDataSet для выполнения SQL-запросов, DataSource в качестве источника данных для других компонентов, DBGrid для вывода результатов SQL запросов.

Для успешного администрирования базы данных следует разделить форму на несколько логических частей. В приложении для администратора будет три вкладки, каждая из которых будет служить для управления одной из частей базы данных.

Форма в приложении для администратора должна иметь три вкладки:

1. Группы и дисциплины;
2. студенты;
3. преподаватели.

Каждая вкладка позволяет добавлять записи, удалять и редактировать записи с соответствующих таблицах базы данных.

Приложение получает информацию о студентах из базы данных. Если не нажимать на ячейки в таблице, то в пустых компонентах Edit можно внести атрибуты в будущую запись и по нажатию соответствующей кнопки добавить нового студента.

Если же нажать на уже существующую запись о студенте, то соответствующие атрибутам компоненты Edit примут значение, равное атрибутам выделенной записи. Кнопка «Добавить» исчезнет, но появятся кнопки «Удалить» и «Изменить».

По нажатию «Изменить», выделенная запись о студенте примет те значения атрибутов, которые стоят на соответствующих значениях компонентов Edit.

По нажатию «Удалить», выделенная запись будет удалена из базы данных. Это возможно, так как при каждом нажатии на записи, будет высчитываться ID записи. Удаление происходит по параметру ID.

Подобные алгоритмы реализованы на каждой вкладке и принцип работы одинаковый. Данное приложение должно облегчить работу сотруднику образовательного учреждения, так как позволяет не использовать инструменты СУБД FireBird и имеет дружественный и понятный интерфейс, а так же обеспечивает корректную работу с базой данных человеку без соотетствующих навыков.

3.2.2. Разработка Windows приложения для управлением расписанием

Для ведения расписания необходимо подготовить компоненты, которые позволят работать с базой данных и отражать расписание учебного заведения.

Известно, что воскресение не является учебным днем, поэтому нужно создать основу для шести дней в неделе. Так же, необходимо обеспечить пользователю возможность изменять неделю, двигаясь вперед или назад по календарю.

Стоить отметить, что в одном учебном дне может быть шесть пар, так же приложение должно обрабатывать случаи, когда в расписании существуют «окна», т.е. пары нет, но следующая будет.

Пользователю необходимо выбрать группу для которой будет редактироваться рассписание. После этого, приложение обращается к базе данных и ищет расписание для этой группы. Если расписания не существует, то все поля ComboBox остаются пусты, иначе поле каждого компонента заполняются данными из базы данных.

В дальнейшем можно отредактироать расписание или изменить. Для облегчения рутины, чтобы не создавать расписание для каждой недели, следует реализовать функцию, которая копирует текущую неделю на все остальные.

Так же в меню должна быть функция по быстрому очищению полей ComboBox, удалению расписанию текущей группы из базы данных.

Приложение для ведения расписания дожно работать корректно и позволит в удобной форме вести учебное расписание человеку, который не имеет достаточных навыков с работой с базой данных. Данное приложение структуирует и визуализирует информацию для удобства пользователя.

3.2.3. Разработка Windows приложения ведения посещаемости студентов

После создания расписания групп, можно переходить непосредственно к ведению посещаемости студентов учебного заведения. Приложение так же должно структуировать и визуализировать расписание группы, взяв необходимую информацию из базы данных. Основной сложностью данного

приложения было то, что невозможно уточнить сколько студентов в группе заранее. Для каждого студента создается свой компонет Label и группа компонентов ComboBox. Было принято решение о инициализации динамических компонетов.

В начале работы приложения не существует заранее готовых компонентов для студентов и ведения их посещаемости. Запустив приложение и выбрав нужную группу и дату, дожны создаться динамические объекты. Если в выбранный день не было пары, то объект ComboBox не создается. Так же, если в выбранный день у группы нет пар, должно выводиться соответствующее сообщение.

Обязательно для приложения ведения посещаемости должна быть реализована функция, которая подгружает информацию из базы данных о ранее сделанных заметках о посещении. Например, если у студента уважительная причина пропуска, то в соответствуюем компоненте Memo может присутстовать причина отсутствия студента.

Данное приложение должно работать исправно и позволять вести посещаемость студентов учебного заведения сотруднику, который не имеет навыков для работы с базой данных. Приложение структуирует и визуализирует информацию из базы данных.

3.2.4. Разработка Web приложения контроля посещаемости для родителей студентов

После реализации приложений, ответственных за внесение информации, необходимо приступить к созданию информационного приложения, которое позволит в свободном доступе обращаться к базе данных и получать сведения

о пропусках. Данное приложение создается для родителей, а значит необходимо, чтобы приложение было доступно на любом устройстве с доступом в интернет. Было принято решение о создании Web-приложения, так как при реализации подобного Windows приложении потребовалось распространить его на устройства всех родителей, в то время, как Web-приложение может запускаться из любого современного браузера.

Как было сказано выше, Web-приложение должно выводить информацию о пропусках. Чтобы это реализовать, необходимо создать главную html страницу, на которой пользователь выбирает дату и студенческую группу. Полученные данные для запроса передаются на вторую страницу, где выводится сводная информация о пропусках всех студентов в группе в течении той недели, в которую входить выбранная дата. Если в текущей неделе отсутствует один или несколько учебных дней, то они должны пропускаться, а не создаваться пустые таблицы для таких дней.

Чтобы было легче считать пропуски конкретного студента, должна быть создана еще одна страница html. На второй странице каждое ФИО студента это ссылка на третью, где считаются все пропуски, как уважительные, так и неуважительные, выбранного студента за каждый месяц в расписании.

Так же стоит добавить функцию, которая позволит пользователю узнавать сопроводительный комментарии к пропуску студента одним лишь наведением мышкой на требуемый пропуск в таблице.

Web-приложение должно иметь понятный интерфейс и не нагружать интернет трафик пользователя.

ГЛАВА 4. ТЕСТИРОВАНИЕ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ КОНТРОЛЯ ПОСЕЩАЕМОСТИ

4.1. Тестирование Windows приложения для администратора

В данном приложение необходимо проверить три важные функции, а именно удаление, изменение и добавление записей в таблицу. Так как принцип работы одинаковый во всех вкладках данного приложения, то тестирование будет проводиться на вкладке «Студенты».

Для начала необходимо проверить функцию вывода уже существующих записей из базы данных. Эта функция выполняется сразу по включению приложения. На рис. 4.1. можно увидеть форму приложения.

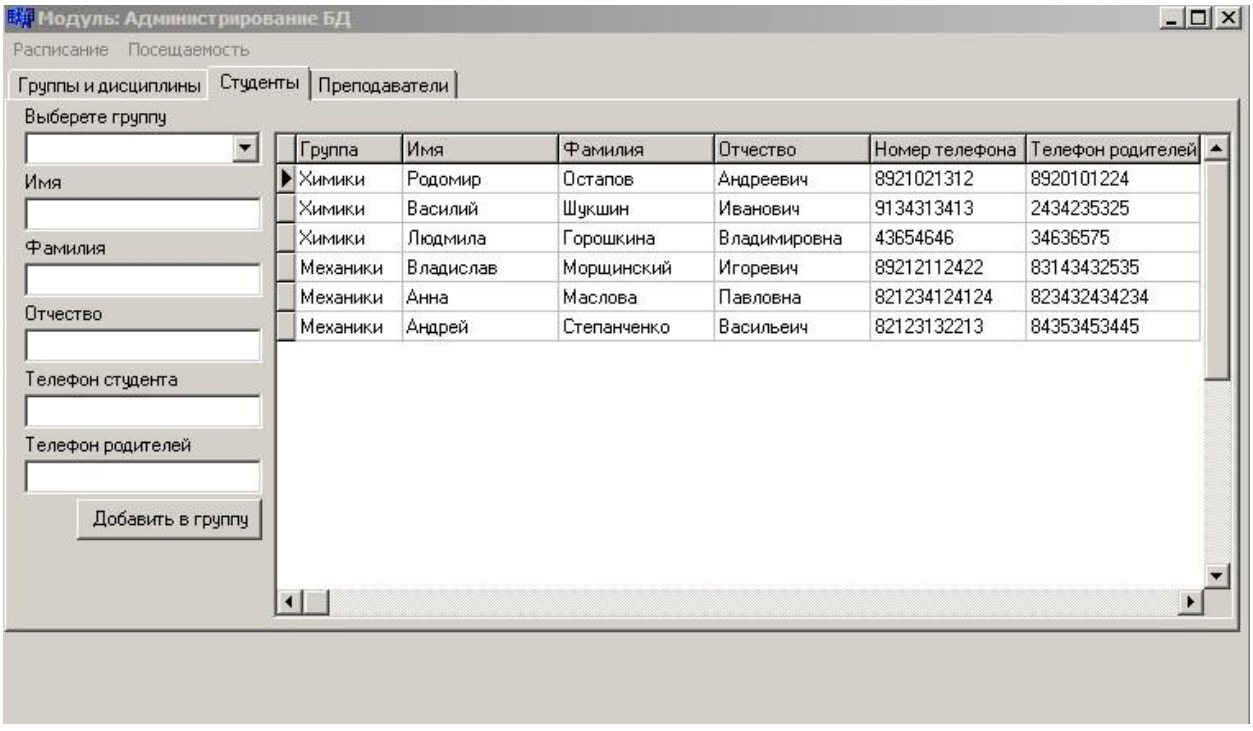


Рис. 4.1. Форма приложения для администратора

На рис. 4.1. видно, что достаточно перейти по вкладке студенты, чтобы информация о существующих студентах была подгружена из базы данных. В

качестве примера вставки записей в базу данных, добавим студента по имени Сапелин Константин Иванович в группу «Механики». Результат добавления студента можно увидеть на рис.4.2.

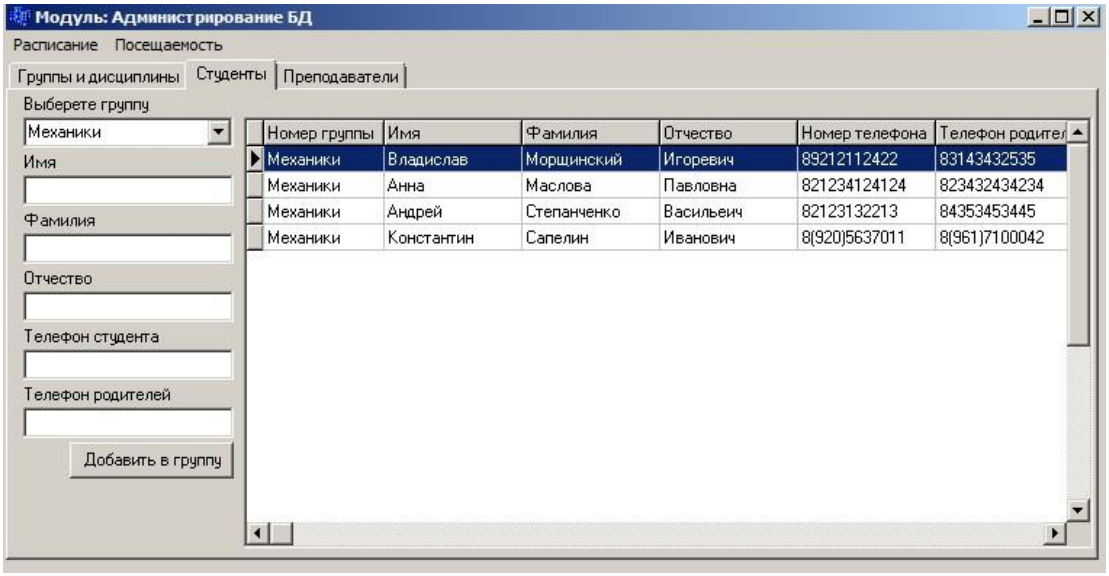


Рис. 4.2. Результат добавления студента в группу «Механики»

Добавление записи прошло успешно. Теперь проведем тестирование функции изменения записи. В качестве примера сделаем так, что студентка поменяла фамилию. В группе «Механики» есть только одна запись одевушке «Анна Маслова». Необходимо сменить ее фамилию на «Лисицина».
Результат изменения записи представлен на рис. 4.3.

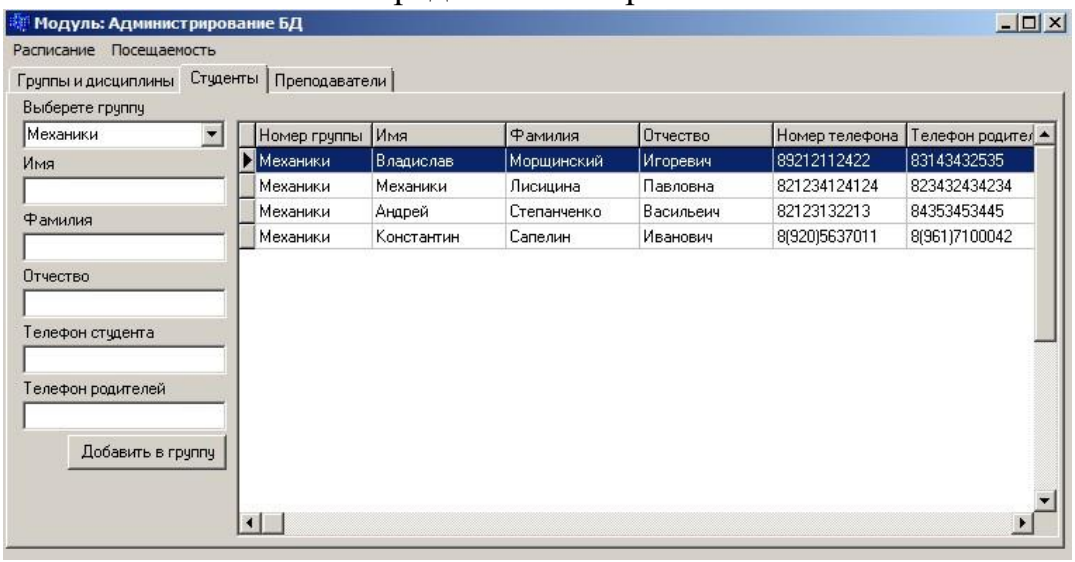


Рис. 4.3. Результат изменения записи в базе данных.

Как видно на рис. 4.3. фамилия студентки изменилась успешно. Дальше необходимо протестировать функцию удаления записи из базы данных. Для этого, в качестве примера удалим запись «Андрей Степанченко». Результат удаления записи приведен на рис. 4.4.

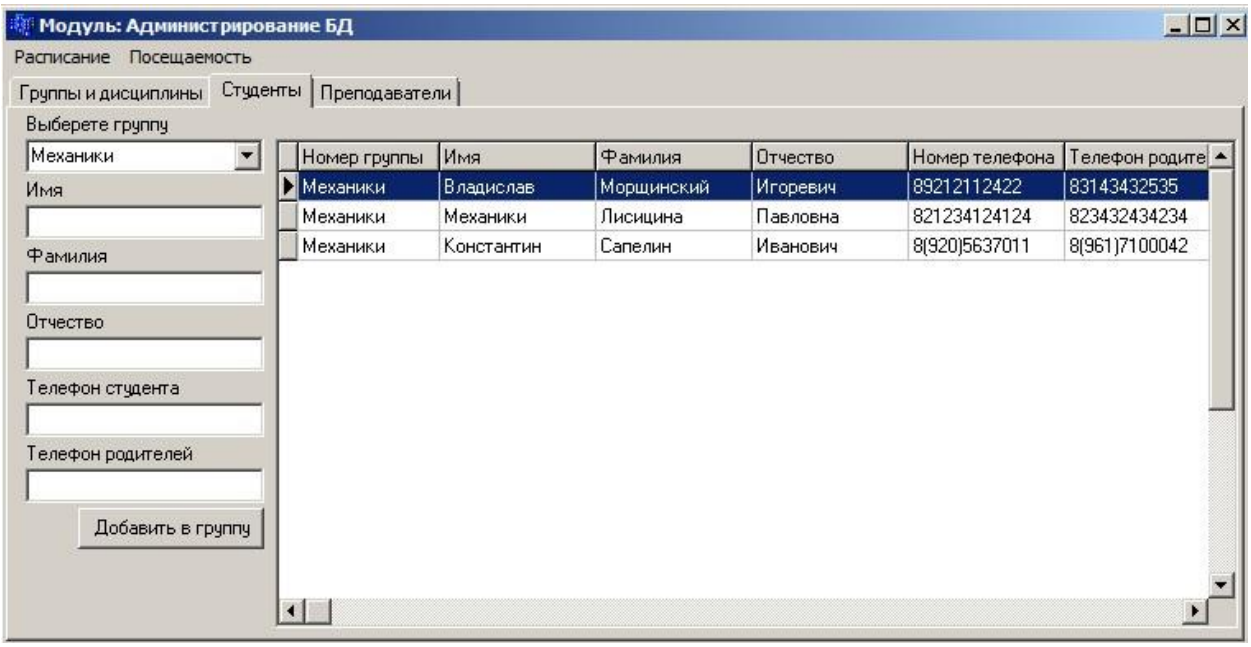


Рис. 4.4. Результат удаления записи из базы данных.

Удаление из базы данных прошло успешно. Так же, при удалении записи о студенте, удаляется и его ведомость посещаемости, т.е. реализовано каскадное удаление из таблиц. Остальные же вкладки и их функции совпадают. Можно считать, что приложение для администрирования базы данных работает корректно. Ошибок при тестировании не обнаружено.

4.2. Тестирование Windows приложения для управления расписанием

Приложения для управления расписанием имеет много компонентов, но визуально структура отражает реальное расписание в неделе. Для

тестирования корректной работы приложения загрузим расписание ранее упомянутой группы «Механики». Результат загрузки расписания, а так же демонстрация формы управления расписания представлен на рис. 4.5.

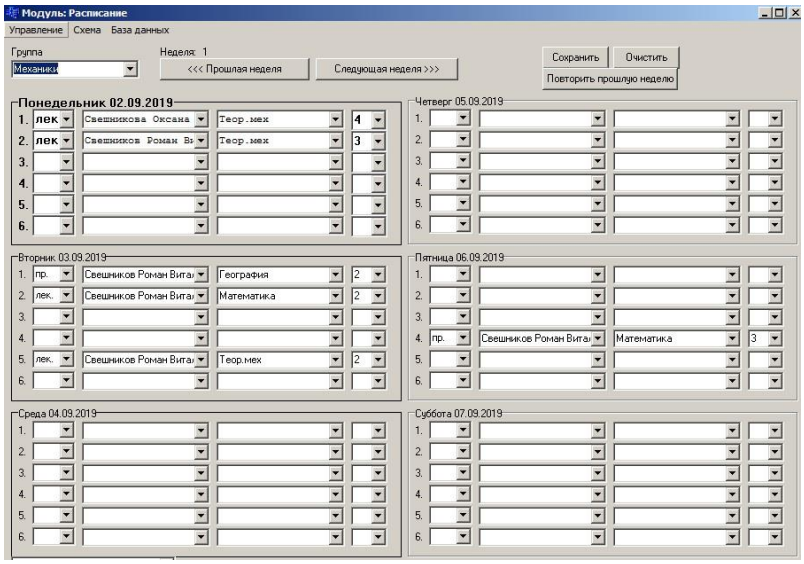


Рис. 4.5. Результат загрузки расписания.

Как видно из рис. 4.5, у группы «Механики» есть пары в понедельник, вторник и пятницу. В качестве примера изменения расписания, перенесем пару с пятницы, на третью пару в понедельник, а так же добавим две пары математики на среду, но только на текущую неделю. Результат теста приведен на рис. 4.6.

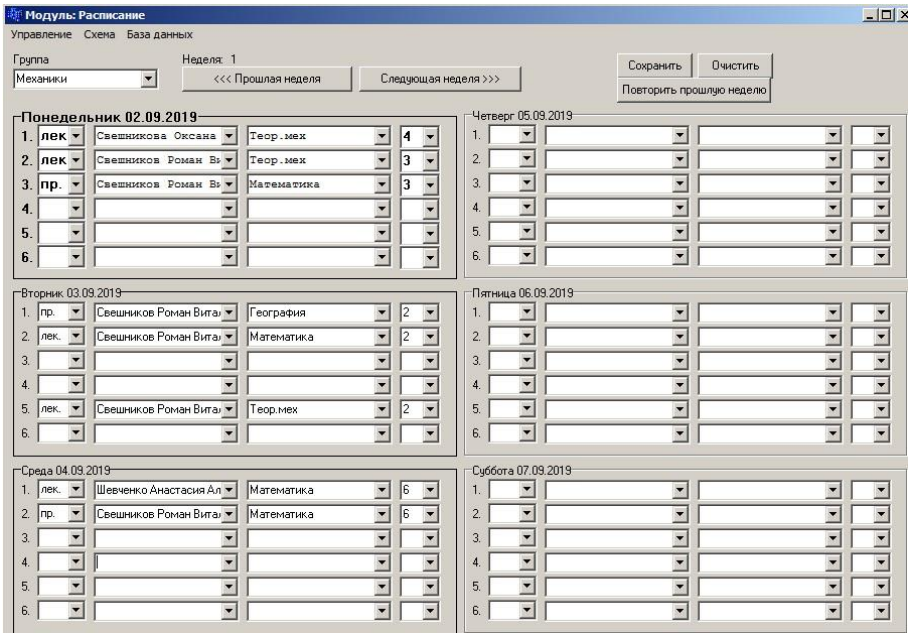


Рис. 4.6. Результат изменения расписания.

Теперь необходимо убедиться, что расписание изменилось только на текущей неделе. Для этого перейдем на следующую неделю. Результат выполнения приведен на рис. 4.7.

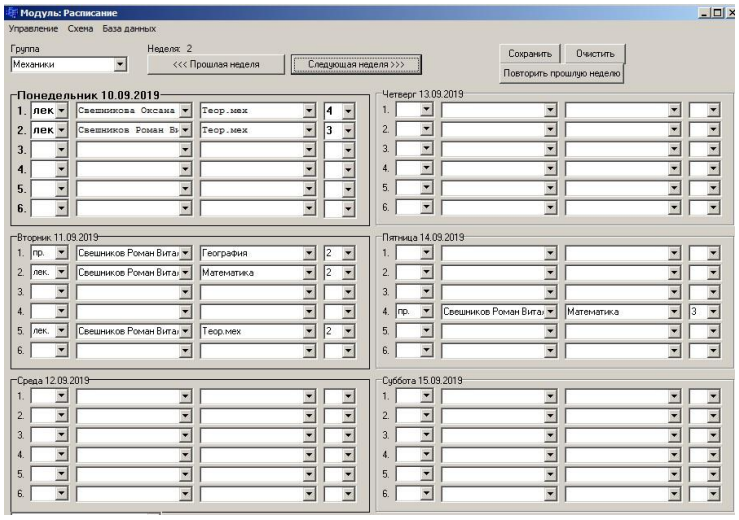


Рис. 4.7. Результат смены недели в приложении.

Можно заметить, что при смене недели так же меняются и даты рядом с соответствующими днями.

Последним, что следует протестировать это возможность скопировать текущую неделю и распространить на все недели. В качестве примера

скопируем первую неделю на все остальные. Пример работы функции и демонстрация расписания на пятой недели приведен на рис. 4.8.

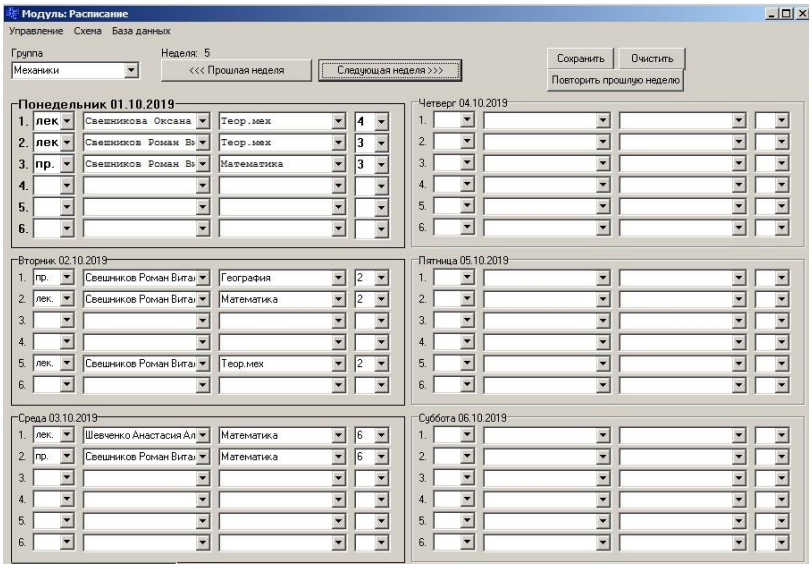


Рис. 4.8. Результат работы функции копирования недели.

Так же приложение имеет функции удаления расписания из базы данных, стирание полей ComboBox. Эти функции реализованы и работают исправно.

В результате тестирования приложения для управлением расписания ошибок было не обнаружено. Приложение работает исправно.

4.3. Тестирование Windows приложения для ведения посещаемости студентов

По существующему расписанию строится ведомость, а по списку студентов из базы данных формируется личная ведомость. Приложению до начала работы не известно количество студентов в группе, поэтому при выборе даты и группы приложение анализирует количество студентов и создает динамические объекты типа TLabel и TComboBox. В качестве примера

запустим приложение, чтобы поставить посещаемость группы «Механики», а дата «01.10.2019». Это понедельник и должно быть три пары.

Результат выполнения примера приведен на рис. 4.9.

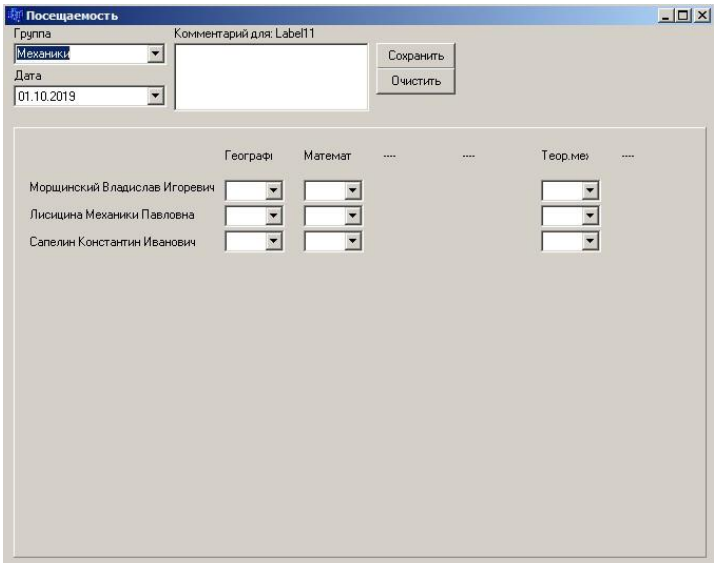


Рис. 4.9. Результат загрузки расписания и студентов из базы данных.

Из рис. 4.9. видно, что расписание загрузилось из базы данных верно. Так же, студентов в группе «Механики» трое, соответственно было создано три набора динамических компонентов для каждого из студентов.

Чтобы протестировать саму функцию ведения посещаемости, проставим уважительные прогулы студенту «Сапелин Константин» всю первую неделю в октябре, а студенту «Лисицина Анна» прогул по неуважительной причине «02.09.2019». Первая неделя октября в 2019 году включает и 30 сентября в понедельник, поэтому у «Сапелин Константин» должен быть прогул и в сентябре. Процесс выставления посещаемости приведен на рис. 4.10.

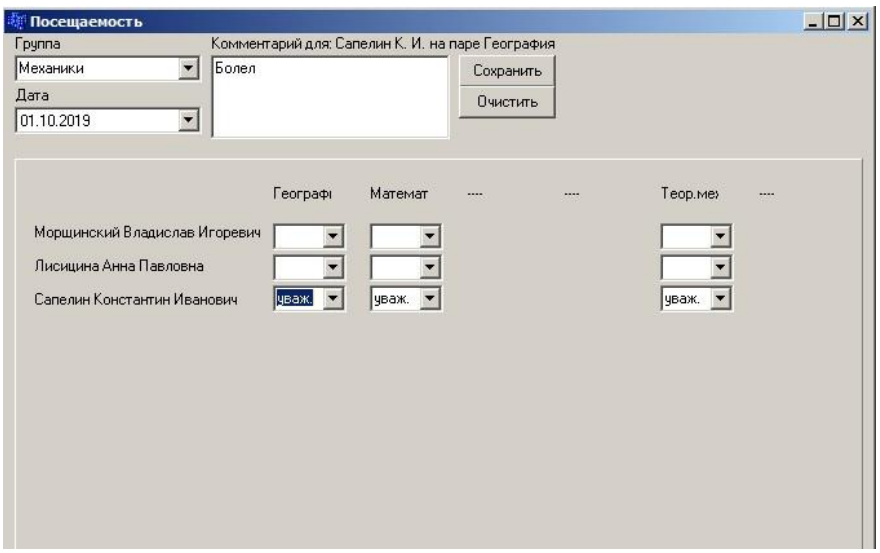


Рис 4.10. Процесс ведения посещаемости студентов

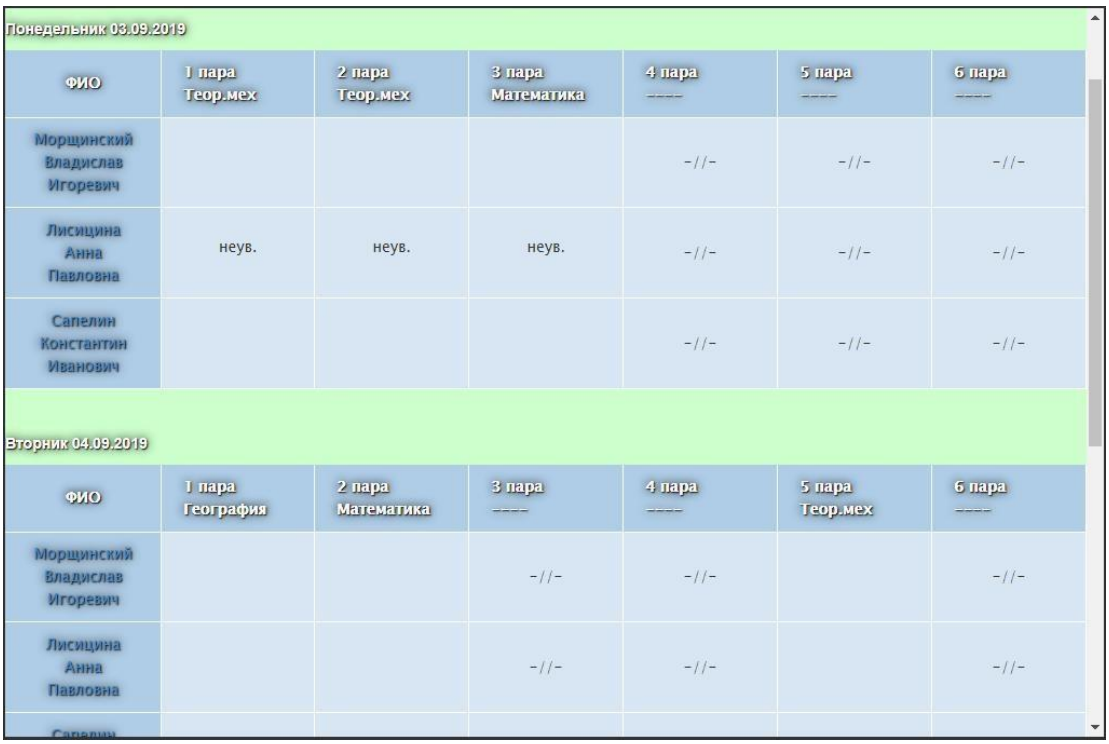
Результат тестирования будет подведен в следующем параграфе, где будет тестироваться уже вывод ведомостей посещаемости студентов образовательного учреждения.

В результате тестирования приложения для ведения посещаемости студентов выявлено ошибок не было. Приложение работает исправно.

4.4. Тестирование Web-приложения контроля посещаемости для родителей студентов

Web-приложение предполагает, что пользоваться им будут родители студентов, либо работники учебной части. В правильности работы приложением можно убедиться, проверив все изменения, которые были сделаны в базе данных в прошлых пунктах в качестве тестирования.

Для начала проверим посещаемость студентов группы «Механики» по дате «02.09.2019». Результат проверки приведен на рис. 4.11.



Понедельник 03.09.2019

ФИО	1 пара Теор.мех	2 пара Теор.мех	3 пара Математика	4 пара -----	5 пара -----	6 пара -----
Морщинский Владислав Игоревич				-//-	-//-	-//-
Лисицина Анна Павловна	неув.	неув.	неув.	-//-	-//-	-//-
Сапелин Константин Иванович				-//-	-//-	-//-

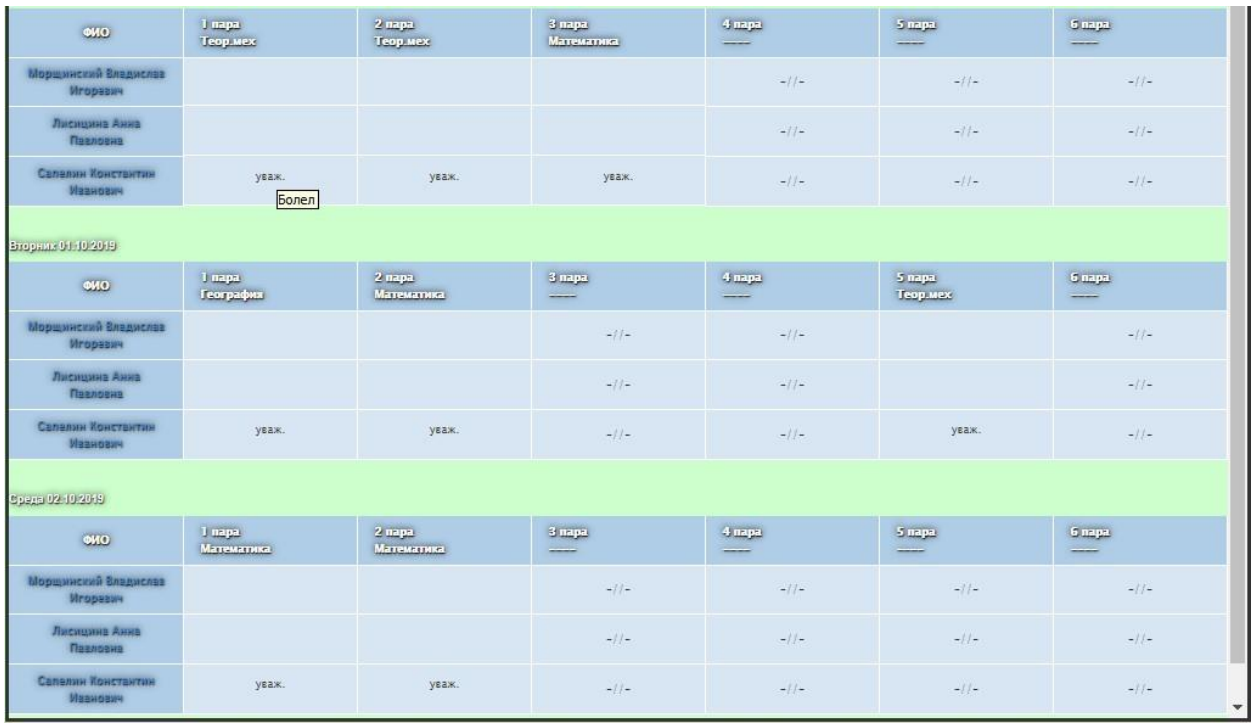
Вторник 04.09.2019

ФИО	1 пара География	2 пара Математика	3 пара ----	4 пара ----	5 пара Теор.мех	6 пара ----
Морщинский Владислав Игоревич			-//-	-//-		-//-
Лисицина Анна Павловна			-//-	-//-		-//-
Сапелин						

Рис. 4.11. Вывод посещаемости студентов в первую неделю сентября

В то время, когда пар нет приложение проставляет знак «-//-», который обозначает окно в расписании группы.

Как видно из рис. 4.11, в прошлом параграфе были проставлены пропуски по неуважительной причине Лисициной Анне в первую неделю сентября. Теперь проверим пропуски студента «Сапелин Константин» в первую неделю октября. Результат этого действия приведен на рис. 4.12.



ФИО	1 пара Теор.мех	2 пара Теор.мех	3 пара Математика	4 пара ----	5 пара ----	6 пара ----
Морщинский Владислав Игоревич				-//-	-//-	-//-
Лисицина Анна Павловна				-//-	-//-	-//-
Сапелин Константин Иванович	уваж. Болел	уваж.	уваж.	-//-	-//-	-//-

Вторник 01.10.2019

ФИО	1 пара География	2 пара Математика	3 пара ----	4 пара ----	5 пара Теор.мех	6 пара ----
Морщинский Владислав Игоревич			-//-	-//-		-//-
Лисицина Анна Павловна			-//-	-//-		-//-
Сапелин Константин Иванович	уваж.	уваж.	-//-	-//-	уваж.	-//-

Среда 02.10.2019

ФИО	1 пара Математика	2 пара Математика	3 пара ----	4 пара ----	5 пара ----	6 пара ----
Морщинский Владислав Игоревич			-//-	-//-	-//-	-//-
Лисицина Анна Павловна			-//-	-//-	-//-	-//-
Сапелин Константин Иванович	уваж.	уваж.	-//-	-//-	-//-	-//-

Рис. 4.12. Результат проверки посещаемости студента

Как видно на рис. 4.12. при наведении на ячейку с пропуском появляется подсказка, которая выводит сопроводительный комментарий к прогулу студента.

Для того, чтобы вывести индивидуальную ведомость посещаемости, необходимо перейти по ссылке, которой является ФИО в таблице. Индивидуальная ведомость суммирует пропуски студента по месяцам, которые есть в расписании группы. Как было сказано выше, студент «Сапелин Константин» пропустил первую неделю октября, но в эту неделю входит и последний день сентября. Значит, что в индивидуальной ведомости должны быть пропуски за октябрь и сентябрь. Индивидуальная ведомость студента приведена на рис. 4.13.

Рис. 4.13. Индивидуальная ведомость студента.

Как видно из рис. 4.13, пропуски поставлены верной с точки зрения календарного плана. Приложение по контролю посещаемости для родителей студентов работает без ошибок и в процессе тестирования не были обнаружены.

ЗАКЛЮЧЕНИЕ

Посещаемость студентов образовательных учреждений это такой же важный аспект образовательного процесса, как зачеты и экзамены. За контролем посещаемости отвечают сотрудники учебной части и их работа сопряжена с постоянным мониторингом. Это не является оптимальным способом регулировки посещаемости студентов, так как зачастую те, кто следит за посещаемостью, сами являются преподавателями и у них есть работа помимо этого.

В Российской Федерации подавляющее большинство образовательных учреждений не имеет автоматизированных систем контроля посещаемости, ни электронных журналов, ни каких других аналогов мониторинга посещаемости, в силу недостатка финансирования. Вместо этого, образовательные

учреждения вынуждены использовать бумажные средства контроля посещаемости или не вести учет вовсе, что ведет к снижению успеваемости.

В случае с автоматизированной системы контроля посещаемости студентов ОГАПОУ “ШТПТ” были реализованы самые необходимые функции для успешного мониторинга посещаемости. Это достигается использование бесплатного программного обеспечения, а так же адаптацией всех реализованных приложений для людей, которые не имеют высоких навыков работы с компьютерными приложениями. Как итог, было реализованы следующие приложения:

• Приложение для администрирования базы данных – служит упрощенным вариантом управления базой данных, так как внутри приложения лишь самые необходимые для этого функции;

• приложение для управлением расписания – объемное приложение и многофункциональное, которое просто необходимо для качественного мониторинга посещаемости студентов образовательных учреждений;

• приложение для ведения посещаемости – приложение для работников учебной части, которое заменяет ручной и письменный труд при контроле посещаемости студентов;

• приложение контроля посещаемости для родителей – конечное звено в мониторинге посещаемости, так как все приложения служат только для внесения данных, в то время как это приложение доносит информацию из базы данных к родителям, сотрудникам учебной части или кураторам.

Так же в процессе выполнения выпускной квалификационной работы были решены поставленные задачи, такие как:

• Рассмотреть порядок ведения контроля посещаемости студентов в Шебекинском техникуме промышленности и транспорта;

• выявить аспекты в порядке ведения контроля посещаемости, которые можно автоматизировать и систематизировать;

• разработать информационную систему, отражающую структуру подразделения по контролю посещаемости;

• реализовать ряд Windows приложений для автоматизации ведения посещаемости;

• реализовать Web-приложение, которое осуществит доступ к информации о посещаемости родителям и педагогам.

В ходе выполнения выпускной квалификационной работы были получены знания и опыт, которые можно в будущем использовать в магистратуре или на рабочем месте.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Кириллов, В.В. Введение в реляционные базы данных.Введение в реляционные базы данных / В.В. Кириллов, Г.Ю. Громов. - СПб.: БХВПетербург, 2012. - 464 стр.

2. Кошелев, В.Е. Базы данных в ACCESS 2007: Эффективное использование / В.Е. Кошелев. - М.: Бином-Пресс, 2009. - 592 стр.

3. Кузин, А.В. Базы данных: Учебное пособие для студ. высш. учеб. заведений / А.В. Кузин, С.В. Левонисова. - М.: ИЦ Академия, 2012. - 320 стр.

4. Ливена, С.В. Практика увольнений за прогул. По материалам базы данных "Пакет кадровика" / С.В. Ливена. - М.: ИНФРА-М, 2008. - 51 стр.

5. Пирогов, В.Ю. Информационные системы и базы данных: организация и проектирование: Учебное пособие / В.Ю. Пирогов. - СПб.: БХВ-Петербург, 2009. - 528 стр.

6. Советов, Б.Я. Базы данных: теория и практика: Учебник для бакалавров / Б.Я. Советов, В.В. Цехановский, В.Д. Чертовской. - М.: Юрайт, 2013.- 463 стр.

7. Фуфаев, Э.В. Базы данных: Учебное пособие для студентов учреждений среднего профессионального образования / Э.В. Фуфаев, Д.Э. Фуфаев. - М.: ИЦ Академия, 2012. - 320 стр.

8. Джон К. ВанДик , Мэт Вестгейт. Pro Drupal 7 Development: Third Edition / Todd Tomlinson . John K. VanDyk - Apress, 2010 - 320 стр.

9. Стивен Хольцнер . PHP в примерах. / Стивен Хольцнер . М.: 000 «Бином-Пресс», 2007 г. Пер. с англ. 352 стр.

10. Ларри Ульман. Ульман Л. Основы программирования на PHP:/Ларри Ульман. Пер. с англ. -М.: ДМК Пресс, 2001. -288 стр.: ил. (Самоучитель).

11. Александр Мазуркевич. MB PHP: настольная книга программиста /Александр Мазуркевич, Дмитрий Еловой. — Мн.: Новое знание, 2003. — 480 стр.: ил.

12. Томсон Лаура. Разработка Web-приложений на PHP и MySQL: Пер. с англ. /Лаура Томсон, Люк Вел.

13. Гутманс Э., Баккен С, Ретанс Д. PHP 5. Профессиональное программирование./ Пер. с англ. СПб: Символ- Плюс, 2006. 704 стр., ил.

14. Довбуш, Галина Visual C++ на примерах / Галина Довбуш , Анатолий Хомоненко. - М.: БХВ-Петербург, 2012. - 528 с.
7. Зиборов, В. MS Visual C++ 2010 в среде .NET / В. Зиборов. - М.: Питер, 2012. - 320 стр.

15. Кетков, Юлий Практика программирования: Visual Basic, C++ Builder, Delphi. Самоучитель / Юлий Кетков , Александр Кетков. - М.: БХВ-Петербург, 2012. - 464 стр.

16. Мешков, А. Visual C++ и MFC / А. Мешков, Ю. Тихомиров. - М.: БХВПетербург, 2013**.** - 546 стр.

17. Неформальное введение в C++ и Turbo Vision. - Москва: ИЛ, 2010. - 384 стр.

18. Панюкова, Т. А. Языки и методы программирования. Создание простых GUI-приложений с помощью Visual C++. Учебное пособие / Т.А. Панюкова, А.В. Панюков. - Москва: Мир, 2015. - 144 стр.

19. Пахомов, Б. C/C++ и MS Visual C++ 2010 для начинающих / Б. Пахомов. - М.: БХВ-Петербург, 2011. - 736 стр.

20. Пахомов, Борис C/C++ и MS Visual C++ 2012 для начинающих / Борис Пахомов. - Москва: СИНТЕГ, 2015. - 518 стр.

21. Пахомов, Борис C/C++ и MS Visual C++ 2012 для начинающих / Борис Пахомов. - М.: "БХВ-Петербург", 2013. - 502 стр.

ПРИЛОЖЕНИЕ 1

Основные функции приложения для администратора

```
void __fastcall TForm1::Button1Click(TObject *Send1er)
{

    DataModule2->IBQuery1->SQL->Text = "INSERT INTO T_GROUP(T_GROUP.NAME)
VALUES (:_name);";
    DataModule2->IBQuery1->Params->ParamByName("_name")->Value = Edit1->Text;
    DataModule2->IBQuery1->ExecSQL();
    DataModule2->IBQuery1->SQL->Text = "";

    DataModule2->T_GROUP->Refresh();
    DBGrid1->Refresh();

    DBGrid1->Columns->Items[0]->Visible = false;
```

```
        DBGrid1->Columns->Items[1]->Title->Caption = "Список групп";

        Edit1->Text = "";

}
//---------------------------------------------------------------------------
void __fastcall TForm1::DBGrid1CellClick(TColumn *Column) {
        String Field, IDField, SQL4ROOM;
int i, count(0), count_room(0);
        Button1->Visible = false;
        Button3->Visible = true;
        Button2->Visible = true;

        IDField = "NAME";
        Field = DBGrid1->DataSource->DataSet->FieldByName(IDField)->AsString;
        Edit1->Text =Field;
        IDField = "ID_GROUP";
        ID = DBGrid1->DataSource->DataSet->FieldByName(IDField)->Value;

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button3Click(TObject *Sender)
{

        DataModule2->IBQuery1->SQL->Text = "UPDATE T_GROUP SET NAME=:_name
WHERE (ID_GROUP =:_id);";
        DataModule2->IBQuery1->Params->ParamByName("_id")->Value = ID;
        DataModule2->IBQuery1->Params->ParamByName("_name")->AsString = Edit1->Text;
        DataModule2->IBQuery1->ExecSQL();
        DataModule2->IBQuery1->SQL->Text = "";

        DataModule2->T_GROUP->Refresh();
        DBGrid1->Refresh();
```

```
        DBGrid1->Columns->Items[0]->Visible = false;
        DBGrid1->Columns->Items[1]->Title->Caption = "Список групп";
        Button1->Visible = true;
        Button3->Visible = false;
        Button2->Visible = false;
        Edit1->Text = "";
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{

            int id_student, count;
            String SQL4GROUP;
            SQL4GROUP = "select COUNT(T_STUDENT.ID_STUDENT) FROM T_STUDENT
WHERE T_STUDENT.ID_GROUP = '"+ IntToStr(ID) +"'";
            DataModule2->IBDataSet4SQL->SelectSQL->Clear();              DataModule2-
>IBDataSet4SQL->SelectSQL->Add(SQL4GROUP);              DataModule2-
>IBDataSet4SQL->Open();              count = DataModule2->DataSource4SQL->DataSet-
>FieldByName("COUNT")>Value;

            SQL4GROUP = "select T_STUDENT.ID_STUDENT FROM T_STUDENT WHERE
T_STUDENT.ID_GROUP = '"+ IntToStr(ID) +"'";
            DataModule2->IBDataSet4SQL->SelectSQL->Clear();
            DataModule2->IBDataSet4SQL->SelectSQL->Add(SQL4GROUP);
            DataModule2->IBDataSet4SQL->Open();
            DataModule2->DataSource4SQL->DataSet->First();
for(int i=0; i<count; i++)
            {
            id_student = DataModule2->DataSource4SQL->DataSet-
>FieldByName("ID_STUDENT")->Value;
            DataModule2->IBQuery1->SQL->Text = "DELETE FROM T_MISS WHERE
ID_STUDENT =:_id";
```

```
        DataModule2->IBQuery1->Params->ParamByName("_id")->Value = id_student;
DataModule2->IBQuery1->ExecSQL();

        DataModule2->DataSource4SQL->DataSet->Next();
        }

        DataModule2->IBQuery1->SQL->Text = "DELETE FROM T_STUDENT WHERE
ID_GROUP =:_id";
        DataModule2->IBQuery1->Params->ParamByName("_id")->Value = ID;
DataModule2->IBQuery1->ExecSQL();

        DataModule2->IBQuery1->SQL->Text = "DELETE FROM T_GROUP WHERE
ID_GROUP =:_id";
        DataModule2->IBQuery1->Params->ParamByName("_id")->Value = ID;
DataModule2->IBQuery1->ExecSQL();

        DataModule2->T_GROUP->Refresh();
        DBGrid1->Refresh();

        DBGrid1->Columns->Items[0]->Visible = false;
        DBGrid1->Columns->Items[1]->Title->Caption = "Список групп";
        Edit1->Text = "";

        Button1->Visible = true;
        Button3->Visible = false;
        Button2->Visible = false;
}
```

ПРИЛОЖЕНИЕ 2

Основные функции приложения для ведения расписания

```
String get_SubString(String input, char split, int element)
      {              int i,j,k,lenght;
char cur_char;              lenght =
input.Length();              String One,
Two, Three, buf;
             i = 1;              j =
1;              buf = "";
cur_char = input[i];
while(i!=lenght)
             {
if(cur_char!=split)
                    {                    buf
= buf + cur_char;
                    }
else                    {
if(j == 3)
                    {
                    Three = buf;
buf = "";
j++;                          }
if(j == 2)
                    {
                    Two = buf;
buf = "";
j++;                          }
if(j == 1)
                    {
                    One = buf;
buf = "";                          j++;
                    }
}              i++;
cur_char = input[i];
```

```
        }

        if(element == 1) {return One;}
if(element == 2) {return Two;}
if(element == 3) {return Three;}
    }

    void get_Type_Lesson(TComboBox *List)
    {
        String SQL4GROUP;
int count_group;

        SQL4GROUP = "select COUNT(*) FROM T_TYPE_LESSON";
        DataModule2->IBDataSet4SQL->SelectSQL->Clear();
        DataModule2->IBDataSet4SQL->SelectSQL->Add(SQL4GROUP);
DataModule2->IBDataSet4SQL->Open();
        count_group    =    DataModule2->DataSource4SQL-
>DataSet>FieldByName("COUNT")->Value;

        SQL4GROUP = "select * FROM T_TYPE_LESSON";
        DataModule2->IBDataSet4SQL->SelectSQL->Clear();
        DataModule2->IBDataSet4SQL->SelectSQL->Add(SQL4GROUP);
DataModule2->IBDataSet4SQL->Open();

        int i;
        List->Items->Clear();
        DataModule2->DataSource4SQL->DataSet->First();
        for(i=0; i<count_group; i++)
        {
            List->Items->Add(DataModule2->DataSource4SQL->DataSet-
>FieldByName("NAME")->Value);
            DataModule2->DataSource4SQL->DataSet->Next();
        }
    };
```

```
TComboBox *mass_combo[6][6][4];


void clear_shedule()
{       int day, lesson,i;
for(day = 0; day < 6; day++)
        {
        for(lesson = 0; lesson < 6; lesson++)
            {                   for(i = 0; i < 4; i++)
mass_combo[day][lesson][i]->Text = "";
            }
        }
}

void clear_all_week()
{
        String SQL4GROUP;
int id_group, id_shedule, count;
        SQL4GROUP = "select T_GROUP.ID_GROUP FROM T_GROUP WHERE
T_GROUP.NAME = '" + ComboBox25->Text +"';";
        DataModule2->IBDataSet4SQL->SelectSQL->Clear();
        DataModule2->IBDataSet4SQL->SelectSQL->Add(SQL4GROUP);
DataModule2->IBDataSet4SQL->Open();
        id_group                =               DataModule2->DataSource4SQL->DataSet-
>FieldByName("ID_GROUP")->Value;

        SQL4GROUP = "SELECT COUNT(t_shedule.ID_SHEDULE) FROM t_shedule
WHERE  t_SHEDULE.id_group ='" + IntToStr(id_group) + "'";
        DataModule2->SHEDULE->SelectSQL->Clear();
        DataModule2->SHEDULE->SelectSQL->Add(SQL4GROUP);
DataModule2->SHEDULE->Open();           count =         DataModule2-
>DataSourceShedule->DataSet->FieldByName("COUNT")>Value;
```

```
        SQL4GROUP = "SELECT t_shedule.ID_SHEDULE FROM t_shedule WHERE t_SHEDULE.id_group ='" + IntToStr(id_group) + "'"; DataModule2->SHEDULE->SelectSQL->Clear();
        DataModule2->SHEDULE->SelectSQL->Add(SQL4GROUP);
        DataModule2->SHEDULE->Open();
        DataModule2->DataSourceShedule->DataSet->First();
for(int i=0; i<count; i++)
            {
            id_shedule = DataModule2->DataSourceShedule->DataSet>FieldByName("ID_SHEDULE")->Value;

            DataModule2->IBQuery1->SQL->Text = "DELETE FROM T_MISS WHERE (T_MISS.ID_SHEDULE =:_id)";
            DataModule2->IBQuery1->Params->ParamByName("_id")->Value = id_shedule;
            DataModule2->IBQuery1->ExecSQL();
            DataModule2->IBQuery1->SQL->Text = "";

            DataModule2->DataSourceShedule->DataSet->Next();
            }

        DataModule2->IBQuery1->SQL->Text = "DELETE FROM T_SHEDULE WHERE (T_SHEDULE.ID_GROUP =:_id)";
        DataModule2->IBQuery1->Params->ParamByName("_id")->Value = id_group;
        DataModule2->IBQuery1->ExecSQL();
        DataModule2->IBQuery1->SQL->Text = "";
        clear_shedule();
    }

    void next_week()
    {
    String Date;
int mod;
    pn = DateTimePicker1->DateTime.DateString();
```

```
        DateTimePicker1->Date = IncDay(DateTimePicker1->Date, 1);
vt = DateTimePicker1->DateTime.DateString();
        DateTimePicker1->Date = IncDay(DateTimePicker1->Date, 1);
sr = DateTimePicker1->DateTime.DateString();
        DateTimePicker1->Date = IncDay(DateTimePicker1->Date, 1);
cht = DateTimePicker1->DateTime.DateString();
        DateTimePicker1->Date = IncDay(DateTimePicker1->Date, 1);
pt = DateTimePicker1->DateTime.DateString();
        DateTimePicker1->Date = IncDay(DateTimePicker1->Date, 1);
sb = DateTimePicker1->DateTime.DateString();
        DateTimePicker1->Date = IncDay(DateTimePicker1->Date, 2);
        }


        void save_shedule(int week)
        {
        String SQL4GROUP, SQL, NAME, FAMILY, DATE;
        int day, para, i, id_group, id_week, id_type_lesson, id_teacher, id_discipline, id_cabinet,
count, id_shedule;
        SQL4GROUP = "select T_GROUP.ID_GROUP FROM T_GROUP WHERE
T_GROUP.NAME = '" + ComboBox25->Text +"';";
        DataModule2->IBDataSet4SQL->SelectSQL->Clear();
        DataModule2->IBDataSet4SQL->SelectSQL->Add(SQL4GROUP);        DataModule2-
>IBDataSet4SQL->Open();
        id_group    =    DataModule2->DataSource4SQL->DataSet->FieldByName("ID_GROUP")-
>Value;        id_week
= week;
SQL4GROUP =
"SELECT
COUNT(T_SHEDUL
E.ID_SHEDULE)
FROM T_SHEDULE

WHERE (ID_GROUP ='" + IntToStr(id_group) + "') and (ID_WEEK = '" + IntToStr(id_week) +
"')";
        DataModule2->IBDataSet4SQL->SelectSQL->Clear();
```

```
DataModule2->IBDataSet4SQL->SelectSQL->Add(SQL4GROUP); DataModule2->IBDataSet4SQL->Open(); count = DataModule2->DataSource4SQL->DataSet->FieldByName("COUNT")->Value;

SQL4GROUP = "SELECT T_SHEDULE.ID_SHEDULE FROM T_SHEDULE WHERE (ID_GROUP ='" + IntToStr(id_group) + "') and (ID_WEEK = '" + IntToStr(id_week) + "')";
DataModule2->IBDataSet4SQL->SelectSQL->Clear();
DataModule2->IBDataSet4SQL->SelectSQL->Add(SQL4GROUP);
DataModule2->IBDataSet4SQL->Open();
DataModule2->DataSource4SQL->DataSet->First();
for(i=0; i<count; i++)
{
    id_shedule = DataModule2->DataSource4SQL->DataSet->FieldByName("ID_SHEDULE")->Value;
    DataModule2->IBQuery1->SQL->Text = "DELETE FROM T_MISS WHERE (ID_SHEDULE =:_id)";
    DataModule2->IBQuery1->Params->ParamByName("_id")->Value = id_shedule;
    DataModule2->IBQuery1->ExecSQL();
    DataModule2->IBQuery1->SQL->Text = "";

    DataModule2->DataSource4SQL->DataSet->Next();
}

DataModule2->IBQuery1->SQL->Text = "DELETE FROM T_SHEDULE WHERE (ID_GROUP =:_id) and (ID_WEEK = :_week)";
DataModule2->IBQuery1->Params->ParamByName("_id")->Value = id_group;
DataModule2->IBQuery1->Params->ParamByName("_week")->Value = id_week;
DataModule2->IBQuery1->ExecSQL();
DataModule2->IBQuery1->SQL->Text = "";
DataModule2->SHEDULE->Active = false;
DataModule2->SHEDULE->Active = true;
```

```
for(day = 0; day < 6; day++)
{
    for(para = 0; para < 6; para++)
    {
        if((mass_combo[day][para][0]->Text=="")||(mass_combo[day][para][1]->Text=="")||(mass_combo[day][para][2]->Text=="")||(mass_combo[day][para][3]->Text==""))
            {
continue;
            }
        SQL = "select T_TYPE_LESSON.ID_TYPE_LESSON FROM T_TYPE_LESSON WHERE T_TYPE_LESSON.NAME = '" + mass_combo[day][para][0]->Text +"';";
        DataModule2->IBDataSet4SQL->SelectSQL->Clear();
        DataModule2->IBDataSet4SQL->SelectSQL->Add(SQL);
DataModule2->IBDataSet4SQL->Open();
        id_type_lesson = DataModule2->DataSource4SQL->DataSet>FieldByName("ID_TYPE_LESSON")->Value;

        FAMILY = get_SubString(mass_combo[day][para][1]->Text,' ', 1);
        NAME = get_SubString(mass_combo[day][para][1]->Text,' ', 2);
        SQL = "select T_TEACHER.ID_TEACHER FROM T_TEACHER WHERE (T_TEACHER.NAME = '" + NAME +"') and (T_TEACHER.FAMILY = '" + FAMILY + "');";
        DataModule2->IBDataSet4SQL->SelectSQL->Clear();
        DataModule2->IBDataSet4SQL->SelectSQL->Add(SQL);
DataModule2->IBDataSet4SQL->Open();
        id_teacher = DataModule2->DataSource4SQL->DataSet>FieldByName("ID_TEACHER")->Value;

        SQL = "select T_DISCIPLINE.ID_DISCIPLINE FROM T_DISCIPLINE WHERE T_DISCIPLINE.NAME = '" + mass_combo[day][para][2]->Text +"';";
DataModule2->IBDataSet4SQL->SelectSQL->Clear();
        DataModule2->IBDataSet4SQL->SelectSQL->Add(SQL);
DataModule2->IBDataSet4SQL->Open();
```

```
id_discipline = DataModule2->DataSource4SQL-
>DataSet>FieldByName("ID_DISCIPLINE")->Value;

SQL = "select T_CABINET.ID_CABINET FROM T_CABINET WHERE
T_CABINET.NAME = '" + mass_combo[day][para][3]->Text +"';";
DataModule2->IBDataSet4SQL->SelectSQL->Clear();
DataModule2->IBDataSet4SQL->SelectSQL->Add(SQL);
DataModule2->IBDataSet4SQL->Open();
id_cabinet = DataModule2->DataSource4SQL-
>DataSet>FieldByName("ID_CABINET")->Value;

DataModule2->IBQuery1->SQL->Text = "INSERT INTO
T_SHEDULE(ID_WEEK, ID_DAY, ID_LESSON, ID_TYPE_LESSON, ID_GROUP,
ID_DISCIPLINE, ID_TEACHER, ID_CABINET, DATKA) VALUES (:_week, :_day, :_lesson,
:_type_lesson, :_group, :_disp, :_teach, :_cab, :_date);";
DataModule2->IBQuery1->Params->ParamByName("_week")->Value =
id_week;
DataModule2->IBQuery1->Params->ParamByName("_day")->Value = day+1;
DataModule2->IBQuery1->Params->ParamByName("_lesson")->Value =
para+1;
DataModule2->IBQuery1->Params->ParamByName("_type_lesson")->Value =
id_type_lesson;
DataModule2->IBQuery1->Params->ParamByName("_group")->Value =
id_group;
DataModule2->IBQuery1->Params->ParamByName("_disp")->Value =
id_discipline;
DataModule2->IBQuery1->Params->ParamByName("_teach")->Value =
id_teacher;
DataModule2->IBQuery1->Params->ParamByName("_cab")->Value =
id_cabinet;
if(day == 0) {DATE = pn;}
if(day == 1) {DATE = vt;} if(day
== 2) {DATE = sr;} if(day == 3)
{DATE = cht;} if(day == 4)
```

```
{DATE = pt;}                    if(day == 5)
{DATE = sb;}
                DataModule2->IBQuery1->Params->ParamByName("_date")->Value = DATE;
                DataModule2->IBQuery1->ExecSQL();
                DataModule2->IBQuery1->SQL->Text = "";
                //DataModule2->IBQuery1->Post();
            }
        }
        next_week();
        DataModule2->IBDataSet4SQL->Transaction->Commit();


        }


        void read_shedule(int week)
        {
        String SQL4GROUP, SQL, NAME, FAMILY, MIDNAME, DATE, TYPE_LESSON,
CABINET, DISCIPLINE;
        int id_day, id_lesson, i, id_group, id_week, id_type_lesson, id_teacher, id_discipline,
id_cabinet, count_shedule;
        SQL4GROUP = "select T_GROUP.ID_GROUP FROM T_GROUP WHERE
T_GROUP.NAME = '" + ComboBox25->Text +"';";
        DataModule2->IBDataSet4SQL->SelectSQL->Clear();
        DataModule2->IBDataSet4SQL->SelectSQL->Add(SQL4GROUP);        DataModule2-
>IBDataSet4SQL->Open();        id_group        =        DataModule2->DataSource4SQL-
>DataSet->FieldByName("ID_GROUP")-
>Value;
        id_week = week;
        SQL4GROUP        =        "select        COUNT(*)        FROM        T_SHEDULE        WHERE
(T_SHEDULE.ID_GROUP = '" + IntToStr(id_group) +"') and (T_SHEDULE.ID_WEEK = '"
+IntToStr(id_week)+ "');";
        DataModule2->SHEDULE->SelectSQL->Clear();
        DataModule2->SHEDULE->SelectSQL->Add(SQL4GROUP);
        DataModule2->SHEDULE->Open();
        //count_shedule        =        DataModule2->SHEDULE->DataSet->FieldByName("COUNT")-
```

```
>Value;  count_shedule = DataModule2->DataSourceShedule->DataSet-
>FieldByName("COUNT")>Value;

    SQL4GROUP = "select * FROM T_SHEDULE WHERE (T_SHEDULE.ID_WEEK = '" +
IntToStr(id_week) + "')";
    DataModule2->SHEDULE->SelectSQL->Clear();
    DataModule2->SHEDULE->SelectSQL->Add(SQL4GROUP);
    DataModule2->SHEDULE->Open();
    DataModule2->DataSourceShedule->DataSet->First();

    for(i=0; i<count_shedule; i++)
    {
        id_day = DataModule2->DataSourceShedule->DataSet->FieldByName("ID_DAY")-
>Value;
        id_lesson = DataModule2->DataSourceShedule->DataSet-
>FieldByName("ID_LESSON")->Value;  id_type_lesson = DataModule2-
>DataSourceShedule->DataSet-
>FieldByName("ID_TYPE_LESSON")->Value;  id_discipline =
  DataModule2->DataSourceShedule->DataSet-
>FieldByName("ID_DISCIPLINE")->Value;  id_teacher = DataModule2-
>DataSourceShedule->DataSet-
>FieldByName("ID_TEACHER")->Value;
        id_cabinet = DataModule2->DataSourceShedule-
>DataSet>FieldByName("ID_CABINET")->Value;

        DataModule2->TYPE_LESSON->SelectSQL->Clear();
        DataModule2->TYPE_LESSON->SelectSQL->Text = "SELECT
T_TYPE_LESSON.NAME FROM T_TYPE_LESSON WHERE
T_TYPE_LESSON.ID_TYPE_LESSON = '"+ IntToStr(id_type_lesson)+ "';";
        DataModule2->TYPE_LESSON->Open();
        TYPE_LESSON = DataModule2->DataSourceTYPE->DataSet-
>FieldByName("NAME")->AsString;  mass_combo[id_day-1][id_lesson-
1][0]->Text = TYPE_LESSON;  DataModule2->TYPE_LESSON-
>SelectSQL->Clear();
```

```
DataModule2->TYPE_LESSON->SelectSQL->Text = "SELECT T_TEACHER.NAME, T_TEACHER.FAMILY, T_TEACHER.MIDNAME FROM T_TEACHER WHERE T_TEACHER.ID_TEACHER = '"+ IntToStr(id_teacher)+ "';";
DataModule2->TYPE_LESSON->Open();
FAMILY = DataModule2->DataSourceTYPE->DataSet->FieldByName("FAMILY")>AsString;
NAME = DataModule2->DataSourceTYPE->DataSet->FieldByName("NAME")->AsString;
MIDNAME = DataModule2->DataSourceTYPE->DataSet->FieldByName("MIDNAME")->AsString;
mass_combo[id_day-1][id_lesson-1][1]->Text = FAMILY +" "+ NAME +" "+ MIDNAME;

DataModule2->TYPE_LESSON->SelectSQL->Clear();
DataModule2->TYPE_LESSON->SelectSQL->Text = "SELECT T_DISCIPLINE.NAME FROM T_DISCIPLINE WHERE T_DISCIPLINE.ID_DISCIPLINE = '"+ IntToStr(id_discipline)+ "';";
DataModule2->TYPE_LESSON->Open();
DISCIPLINE = DataModule2->DataSourceTYPE->DataSet->FieldByName("NAME")->AsString; mass_combo[id_day-1][id_lesson-1][2]->Text = DISCIPLINE;

DataModule2->TYPE_LESSON->SelectSQL->Clear();
DataModule2->TYPE_LESSON->SelectSQL->Text = "SELECT T_CABINET.NAME FROM T_CABINET WHERE T_CABINET.ID_CABINET = '"+ IntToStr(id_cabinet)+ "';";
DataModule2->TYPE_LESSON->Open();
CABINET = DataModule2->DataSourceTYPE->DataSet->FieldByName("NAME")->AsString;
mass_combo[id_day-1][id_lesson-1][3]->Text = CABINET;

DataModule2->DataSourceShedule->DataSet->Next();
}
};
```

```
	void get_Cabinet(TComboBox *List)
	{
		String SQL4GROUP;
int count_group;

		SQL4GROUP = "select COUNT(*) FROM T_CABINET";
		DataModule2->IBDataSet4SQL->SelectSQL->Clear();
		DataModule2->IBDataSet4SQL->SelectSQL->Add(SQL4GROUP);
DataModule2->IBDataSet4SQL->Open();
		count_group	=	DataModule2->DataSource4SQL-
>DataSet>FieldByName("COUNT")->Value;

		int i;
		List->Items->Clear();
		DataModule2->DataSourceCabinet->DataSet->First();
for(i=0; i<count_group; i++)
		{
			List->Items->Add(DataModule2->DataSourceCabinet->DataSet-
>FieldByName("NAME")->Value);
			DataModule2->DataSourceCabinet->DataSet->Next();
		}
	};

	void get_Teacher(TComboBox *List)
	{
		String SQL4GROUP, NAME, FAMILY, MIDNAME, Sum;
int count_group;

		SQL4GROUP = "select COUNT(*) FROM T_TEACHER";
		DataModule2->IBDataSet4SQL->SelectSQL->Clear();
		DataModule2->IBDataSet4SQL->SelectSQL->Add(SQL4GROUP);
		DataModule2->IBDataSet4SQL->Open();
		count_group	=	DataModule2->DataSource4SQL-
>DataSet>FieldByName("COUNT")->Value;
```

```
        int i;
        List->Items->Clear();
        DataModule2->DataSourceTeacher->DataSet->First();
for(i=0; i<count_group; i++)
        {
            FAMILY = DataModule2->DataSourceTeacher->DataSet-
>FieldByName("FAMILY")->Value;
            NAME = DataModule2->DataSourceTeacher->DataSet-
>FieldByName("NAME")->Value;
            MIDNAME = DataModule2->DataSourceTeacher->DataSet-
>FieldByName("MIDNAME")->Value;
            Sum = FAMILY + " " + NAME + " " + MIDNAME;
            List->Items->Add(Sum);
            DataModule2->DataSourceTeacher->DataSet->Next();
        }
    };

    void get_Discipline(TComboBox *List)
    {
        String SQL4GROUP;
int count_group;

        SQL4GROUP = "select COUNT(*) FROM T_DISCIPLINE";
        DataModule2->IBDataSet4SQL->SelectSQL->Clear();
        DataModule2->IBDataSet4SQL->SelectSQL->Add(SQL4GROUP);
DataModule2->IBDataSet4SQL->Open();
        count_group = DataModule2->DataSource4SQL-
>DataSet>FieldByName("COUNT")->Value;
         int
i;
List->Items-
>Clear();
```

```
            DataModule2->DataSourceDISCIPLINE->DataSet->First();
for(i=0; i<count_group; i++)
            {
                List->Items->Add(DataModule2->DataSourceDISCIPLINE->DataSet-
>FieldByName("NAME")->Value);
                DataModule2->DataSourceDISCIPLINE->DataSet->Next();
            }
      };
```

ПРИЛОЖЕНИЕ 3

Листинг Web-приложения

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="css/bootstrap.min.css">
<link rel="stylesheet" type="text/css" href="css/bootstrap-responsive.min.css">
<link rel="stylesheet" type="text/css" href="css/Style.css">
</head>
<body align=center>
        <div class = "main_form">
                <form action="show_shedule.php" method="post"><br>

                <?php
            $id_student = $_GET['id'];              echo '<h1 class = "shadow">
Индивидуальная ведомость </h1><br>';

                $dsn = 'firebird:host=localhost; dbname=C:\DIP.FDB;';
                $user= 'SYSDBA';
                $password = 'masterkey';
                $dbn = new PDO($dsn, $user, $password);
                if(!$dbn) echo "Ошибка подключения! ПРоверьте параметры<br>";

                $sql="select T_STUDENT.FAMILY, T_STUDENT.NAME,
T_STUDENT.MIDNAME, T_STUDENT.ID_STUDENT, T_STUDENT.ID_GROUP FROM
T_STUDENT WHERE T_STUDENT.ID_STUDENT = '".$id_student."'";
                        $res=$dbn->prepare($sql);
                        $res->execute();
                    $result=$res->fetchall(PDO::FETCH_ASSOC);
    foreach($result as $row)
                        {
                                $id_student = $row[ID_STUDENT];
                                $family = $row[FAMILY];
```

```
                $name = $row[NAME];
                $midname = $row[MIDNAME];
                $id_group = $row[ID_GROUP];

                $full_name = $family.' '.$name.' '.$midname;
            }
                echo "<p align=left class =shadow > <b> Ф.И.О. студента - "; echo
$full_name; echo "</b> <p>";
            $arr_miss_bad = array();
    $arr_miss_good = array();                for ($week = 1;
$week <= 18; $week++)
            {
                for ($day = 1; $day <= 6; $day++)
                {

                    $sql="select COUNT(T_SHEDULE.ID_SHEDULE)
FROM T_SHEDULE WHERE (T_SHEDULE.ID_WEEK = '".$week."') AND
(T_SHEDULE.ID_DAY = '".$day."') AND (T_SHEDULE.ID_GROUP = '".$id_group."')";
                    $res=$dbn->prepare($sql);
                    $res->execute();
                    $result=$res->fetchall(PDO::FETCH_ASSOC);

                    foreach($result as $row) {$count = $row[COUNT];}
                    if($count == 0)
                    {

                    }
                    else
                    {
                        for ($para = 1; $para <= 6; $para++)
                        {
                            $sql="select
COUNT(T_SHEDULE.ID_SHEDULE) FROM T_SHEDULE WHERE
```

```
(T_SHEDULE.ID_WEEK = '".$week."') AND (T_SHEDULE.ID_DAY = '".$day."') AND
(T_SHEDULE.ID_LESSON = '".$para."') AND (T_SHEDULE.ID_GROUP = '".$id_group."')";
                                                $res=$dbn->prepare($sql);
                                                $res->execute();
                                                $result=$res-
>fetchall(PDO::FETCH_ASSOC);
                                                        foreach($result as $row) {$count_para =
$row[COUNT];}
                                                if($count_para == 0)
                                                {

                                                }
                                                else
                                                {
                                                        $sql="select
T_SHEDULE.ID_SHEDULE, T_SHEDULE.DATKA FROM T_SHEDULE WHERE
(T_SHEDULE.ID_WEEK = '".$week."') AND (T_SHEDULE.ID_DAY = '".$day."') AND
(T_SHEDULE.ID_LESSON = '".$para."') AND (T_SHEDULE.ID_GROUP = '".$id_group."')";
                                                        $res=$dbn->prepare($sql);
                                                        $res->execute();
                                                        $result=$res-
>fetchall(PDO::FETCH_ASSOC);
                                                         foreach($result as $row) {$id_shedule
= $row[ID_SHEDULE]; $dat = $row[DATKA];}

                                                        $sql="select
COUNT(T_MISS.NAME) FROM T_MISS WHERE (T_MISS.ID_SHEDULE =
'".$id_shedule."') AND (T_MISS.ID_STUDENT = '".$id_student."')";
                                                        $res=$dbn->prepare($sql);
                                                        $res->execute();
                                                        $result=$res-
>fetchall(PDO::FETCH_ASSOC);
                                                        foreach($result as $row)
{$count_miss = $row[COUNT]; }
```

```
                                        if($count_miss == 0)
                                        {

                                        }
                                        else
                                        {
                                            $sql="select T_MISS.NAME
FROM T_MISS WHERE (T_MISS.ID_SHEDULE = '".$id_shedule."') AND
(T_MISS.ID_STUDENT = '".$id_student."')";
                                            $res=$dbn->prepare($sql);
                                            $res->execute();
                                            $result=$res-
>fetchall(PDO::FETCH_ASSOC);
                                            foreach($result as $row)
{$miss = $row[NAME];}

                                            $month = substr($dat, 3, 2);
                                            $nn = 0;
                                            switch ($month)
                                            {
                                                case '09':
                                    $nn = 1;
                                        break;
                                case '10':
                $nn = 2;
        break;                                                  case
'11':                                                       $nn = 3;
                                    break;
                                case '12':
                $nn = 4;
        break;
        case '01':
                                            $nn = 5;
                                    break;
```

```
                                                                    }

                                                                    if($miss == 'уваж.')
                                                                    {
                                                                            $arr_miss_good[$nn] =
$arr_miss_good[$nn] + 1;
                                                                    }
                                                                    if($miss == 'неув.')
                                                                    {
                                                                             $arr_miss_bad[$nn] =
$arr_miss_bad[$nn] + 1;
                                                                    }

                                                            }
                                                        }
                                                    }
                                                }
                                            }
                                        }

  echo '<table border=1 align = center>';    echo '<tr>' .
       '<th bgcolor = "#9d83e6"> Месяц </th>' .
       '<th bgcolor = "#9d83e6"> Кол-во уваж. пропусков </th>' .
       '<th bgcolor = "#9d83e6"> Кол-во неув. пропусков </th>' .
       '<tr>';

       for ($i = 1; $i <= 5; $i++)
                                {
                        switch ($i)
                                    {
            case 1:
            $mn = 'Сентябрь';
    break;
    case 2:
```

```
            $mn = 'Октябрь';
    break;
    case 3:
                $mn = 'Ноябрь';
            break;
    case 4:
            $mn = 'Декабрь';
    break;
    case 5:
            $mn = 'Январь';
    break;
                                    }


                                echo "<tr><td>"; echo $mn; echo "</td>";
            echo "<td>"; echo $arr_miss_good[$i]; echo "</td>";
            echo "<td>"; echo $arr_miss_bad[$i]; echo "</td>";
                                    echo "</tr>";
        }          echo "</table> <br>
<br>";
                    ?>


                    </form>
          </div>
<body>
```